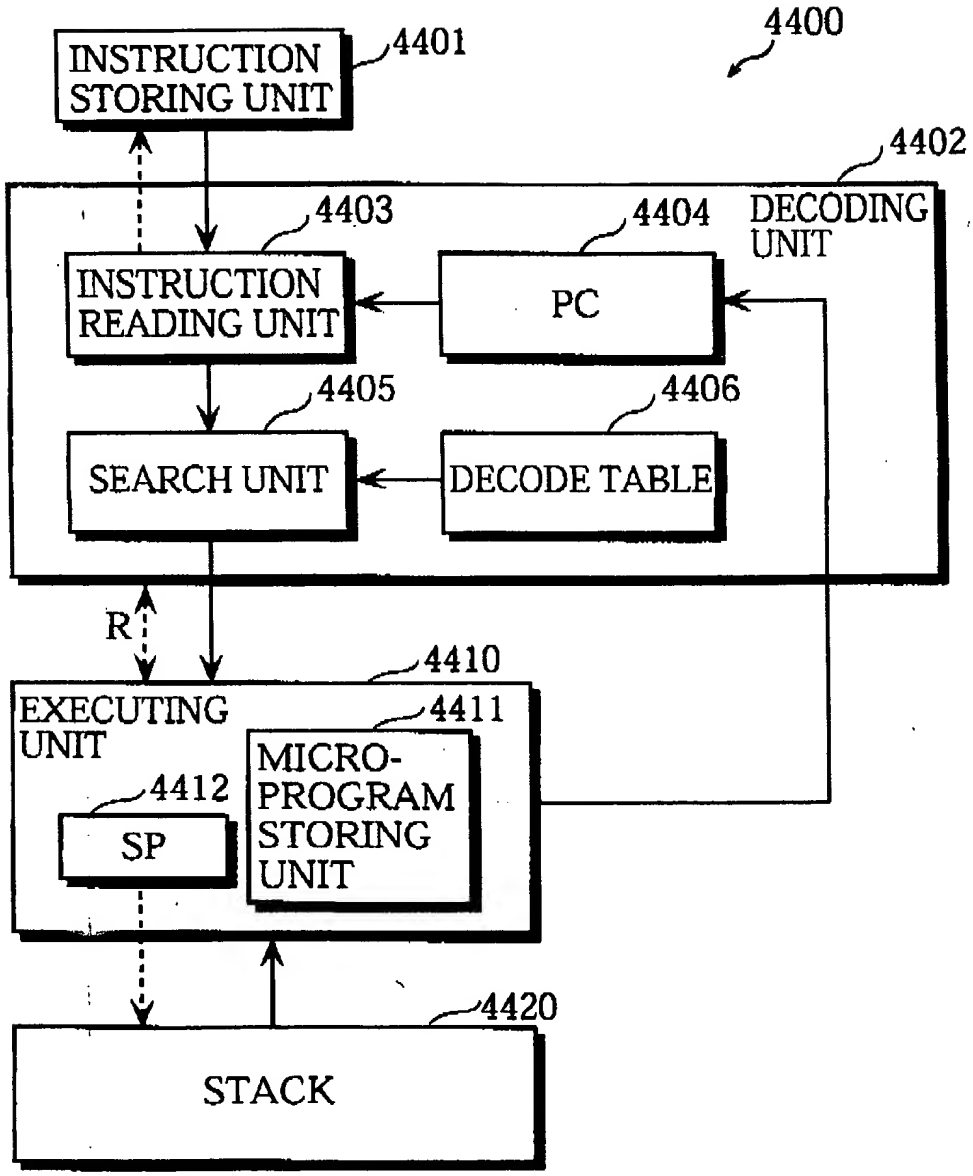
FIG. 1


4400
INSTRUCTION STORING UNIT
4401
4402
DECODING UNIT
4403
INSTRUCTION READING UNIT
4404
PC
4405
SEARCH UNIT
4406
DECODE TABLE
R
4410
EXECUTING UNIT
4411
MICRO-PROGRAM STORING UNIT
4412
SP
4420
STACK

6680h0 E9268260

FIG. 2

VIRTUAL MACHINE INSTRUCTION	OPERATION CONTENTS	CHANGE IN STACK CONTENTS	SP VALUE
Push	PUSH OPERAND ONTO STACK	s0←operand	sp←sp+1
Pop	POP VALUE OF TOP STACK AND PLACE IT INTO ADDRESS INDICATED BY OPERAND	operand←s0 s0←s1	sp←sp-1
Add	ADD VALUES ON TOP AND SECOND STACKS AND STORE RESULT ONTO STACK TOP	s0←s0+s1	sp←sp-1
Mult	MULTIPLY VALUES ON TOP AND SECOND STACKS AND STORE RESULT ONTO STACK TOP	s0←s0*s1	sp←sp-1
Br	JUMP UNCONDITIONALLY TO ADDRESS OF OPERAND	s0←s0	sp←sp
Brz	JUMP TO ADDRESS OF OPERAND IF STACK TOP VALUE IS 0	DELETE←s0 s0←s1	sp←sp-1
Brnz	JUMP TO ADDRESS OF OPERAND IF STACK TOP VALUE IS NOT 0	DELETE←s0 s0←s1	sp←sp-1
Call	CALL FUNCTION SPECIFIED BY ADDRESS OF OPERAND	s0←NEXT VIRTUAL MACHINE CODE ADDRESS	sp←sp+1
Ret	JUMP UNCONDITIONALLY TO ADDRESS OF STACK TOP VALUE	DELETE←s0 s0←s1	sp←sp-1
Stop	STOP VIRTUAL MACHINE PROCESSING	INITIAL STATE	sp←0

FIG. 3

4406

4406a OPCODE	4406b JUMP ADDRESS	4406c NUMBER OF OPERANDS
:	:	:
Push	<JUMP ADDRESS OF CODE TO PERFORM Push>	1
Pop	<JUMP ADDRESS OF CODE TO PERFORM Pop>	1
Add	<JUMP ADDRESS OF CODE TO PERFORM Add>	0
Sub	<JUMP ADDRESS OF CODE TO PERFORM Sub>	0
Inc	<JUMP ADDRESS OF CODE TO PERFORM Inc>	0
Dec	<JUMP ADDRESS OF CODE TO PERFORM Dec>	0
Mult	<JUMP ADDRESS OF CODE TO PERFORM Mult>	0
Div	<JUMP ADDRESS OF CODE TO PERFORM Div>	0
:	:	:

FIG. 4A

```
MICROPROGRAM FOR VIRTUAL MACHINE INSTRUCTION "Push"
1:Inc      r3       ; INCREMENT SP VALUE BY ONE
2:Load     r0,[r2]  ; EXTRACT OPERAND AND
                    ; PLACE IT ONTO REGISTER #0
3:Inc      r2       ; INCREMENT VIRTUAL MACHINE PC BY ONE AND
                    ; PREPARE FOR READING NEXT INSTRUCTION
4:Store    [r3],r0  ; PUSH VALUE OF REGISTER #0 ONTO STACK
<MICROPROGRAM FOR JUMPING TO NEXT VIRTUAL MACHINE INSTRUCTION>
```

FIG. 4B

```
MICROPROGRAM FOR VIRTUAL MACHINE INSTRUCTION "Add"
1:Load     r0,[r3]  ; EXTRACT VALUE FROM STACK
                    ; PLACE IT ONTO REGISTER #0
2:Dec      r3       ; DECREMENT VALUE OF VIRTUAL MACHINE SP BY ONE
3:Load     r1,[r3]  ; EXTRACT VALUE FROM STACK
                    ; PLACE IT ONTO REGISTER #1
4:Add      r0,r0,r1 ; ADD VALUES OF REGISTER #0 AND #1 AND
                    ; PLACE RESULT ONTO REGISTER #1
5:Store    [r3],r0  ; PLACE VALUE OF REGISTER #0 ONTO STACK
<MICROPROGRAM FOR JUMPING TO NEXT VIRTUAL MACHINE INSTRUCTION>
```

FIG. 4C

```
MICROPROGRAM FOR VIRTUAL MACHINE INSTRUCTION "Mult"
1:Load     r0,[r3]  ; EXTRACT VALUE FROM STACK AND
                    ; PLACE IT ONTO REGISTER #0
2:Dec      r3       ; DECREMENT VALUE OF VIRTUAL MACHINE SP BY ONE
3:Load     r1,[r3]  ; EXTRACT VALUE FROM STACK AND
                    ; PLACE IT ONTO REGISTER #1
4:Mult     r0,r0,r1 ; MULTIPLY VALUES OF REGISTERS #0 AND #1 AND
                    ; PLACE RESULT ONTO REGISTER #1
5:Store    [r3],r0  ; PLACE VELUE OF REGISTER #0 ONTO STACK
<MICROPROGRAM FOR JUMPING TO NEXT VIRTUAL MACHINE INSTRUCTION>
```

FIG. 4D

```
<MICROPROGRAM FOR JUMPING TO NEXT VIRTUAL MACHINE INSTRUCTION>
1:Load     r0,[r2]  ; READ VIRTUAL MACHINE INSTRUCTION
                      (JUMP ADDRESS) INDICATED BY PC
                      INTO REGISTER #0
2:Inc      r2       ; INCREMENT VIRTUAL MACHINE PC VALUE BY ONE
3:Jmp      r0       ; JUMP UNCONDITIONALLY TO LOCATION
                      INDICATED BY REGISTER #0
```

6680n0 E928826O

FIG. 5

REAL MACHINE INSTRUCTION	OPERATION CONTENTS	NOTATION	EXAMPLE
Load	PLACE VALUE OF DESIGNATED REGISTER (x1) OR THAT OF MEMORY LOCATION ([x1]) DESIGNATED BY REGISTER ONTO VALUE OF DESIGNATED REGISTER (x0)	Load x0,x1 Load x0,[x1]	Load r0,r1 Load r1,[r2]
Store	PLACE VALUE OF DESIGNATED REGISTER (x1) ONTO MEMORY LOCATION [x0] DESIGNATED BY REGISTER	Store [x0],x1	Store [r0],r1
Add	ADD VALUES OF DESIGNATED REGISTERS (x1,x2) , AND PLACE RESULT ONTO DESIGNATED REGISTER (x0)	Add x0,x1,x2	Add r0,r1,r2
Mult	MULTIPLY VALUES OF DESIGNATED REGISTERS (x1,x2) , AND PLACE RESULT ONTO DESIGNATED REGISTER (x0)	Mult x0,x1,x2	Mult r0,r1,r2
Inc	ADD 1 TO VALUE OF DESIGNATED REGISTER (x1) , AND STORE RESULT IN SAME REGISTER	Inc x0	Inc r4
Dec	SUBTRACT 1 FROM VALUE OF DESIGNATED REGISTER (x0) , AND STORE RESULT IN SAME REGISTER	Dec x0	Dec r3
Jmp	JUMP TO ADDRESS INDICATED BY DESIGNATED REGISTER (x0)	Jmp x0	Jmp r2
Jz	JUMP TO ADDRESS INDICATED BY DESIGNATED REGISTER (x1) IF VALUE OF DESIGNATED REGISTER (x0) IS 0	Jz x0,x1	Jz r0,r2
Jnz	JUMP TO ADDRESS INDICATED BY DESIGNATED REGISTER (x1) IF VALUE OF DESIGNATED REGISTER (x0) IS NOT 0	Jnz x0,x1	Jnz r0,r2
Nop	NOTHING PERFORMED	Nop	Nop

FIG. 6

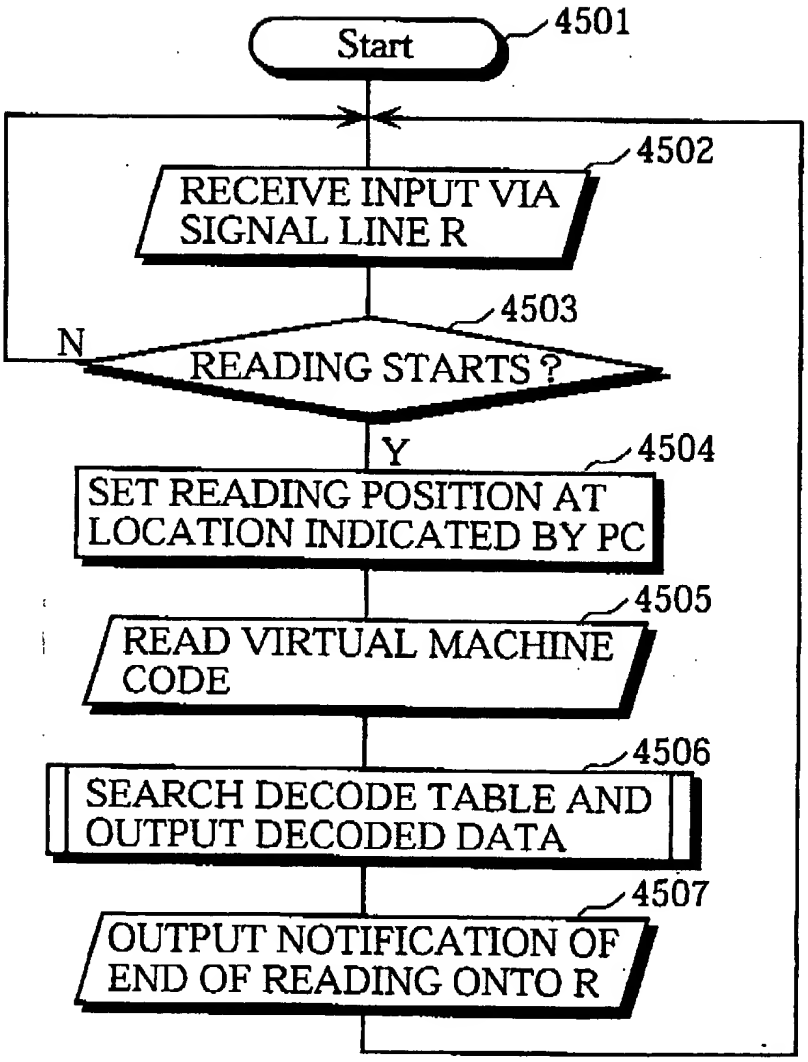

Start
4501
RECEIVE INPUT VIA SIGNAL LINE R
4502
READING STARTS ?
4503
N
Y
SET READING POSITION AT LOCATION INDICATED BY PC
4504
READ VIRTUAL MACHINE CODE
4505
SEARCH DECODE TABLE AND OUTPUT DECODED DATA
4506
OUTPUT NOTIFICATION OF END OF READING ONTO R
4507

FIG. 7

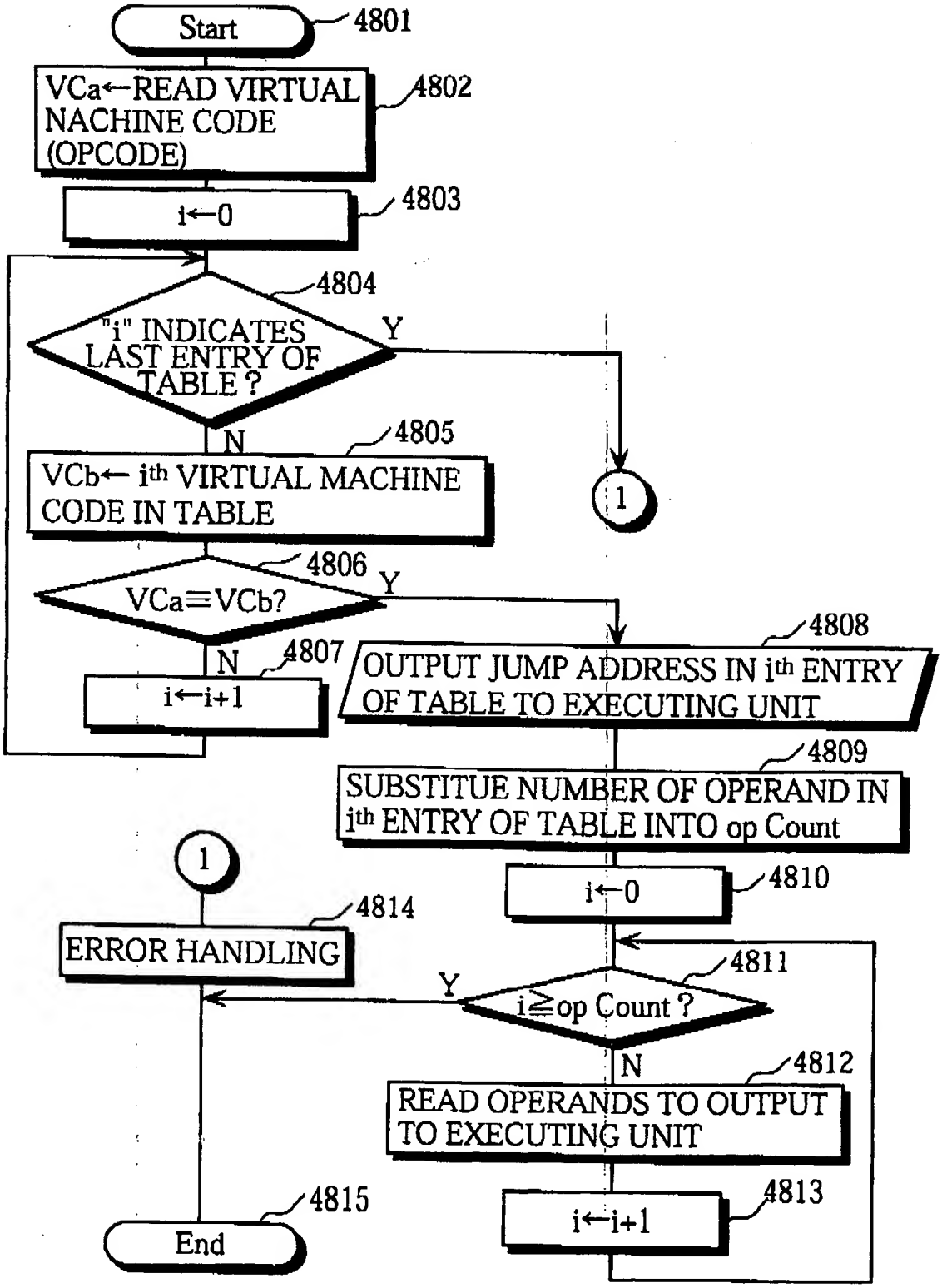

Start
4801
VCa←READ VIRTUAL NACHINE CODE (OPCODE)
4802
i←0
4803
4804
"i" INDICATES LAST ENTRY OF TABLE ?
Y
N
4805
VCb← ith VIRTUAL MACHINE CODE IN TABLE
1
4806
VCa≡VCb?
Y
N
4807
i←i+1
4808
OUTPUT JUMP ADDRESS IN ith ENTRY OF TABLE TO EXECUTING UNIT
4809
SUBSTITUE NUMBER OF OPERAND IN ith ENTRY OF TABLE INTO op Count
4810
i←0
1
4814
ERROR HANDLING
4811
Y
i≧op Count ?
N
4812
READ OPERANDS TO OUTPUT TO EXECUTING UNIT
4813
i←i+1
4815
End

FIG. 8

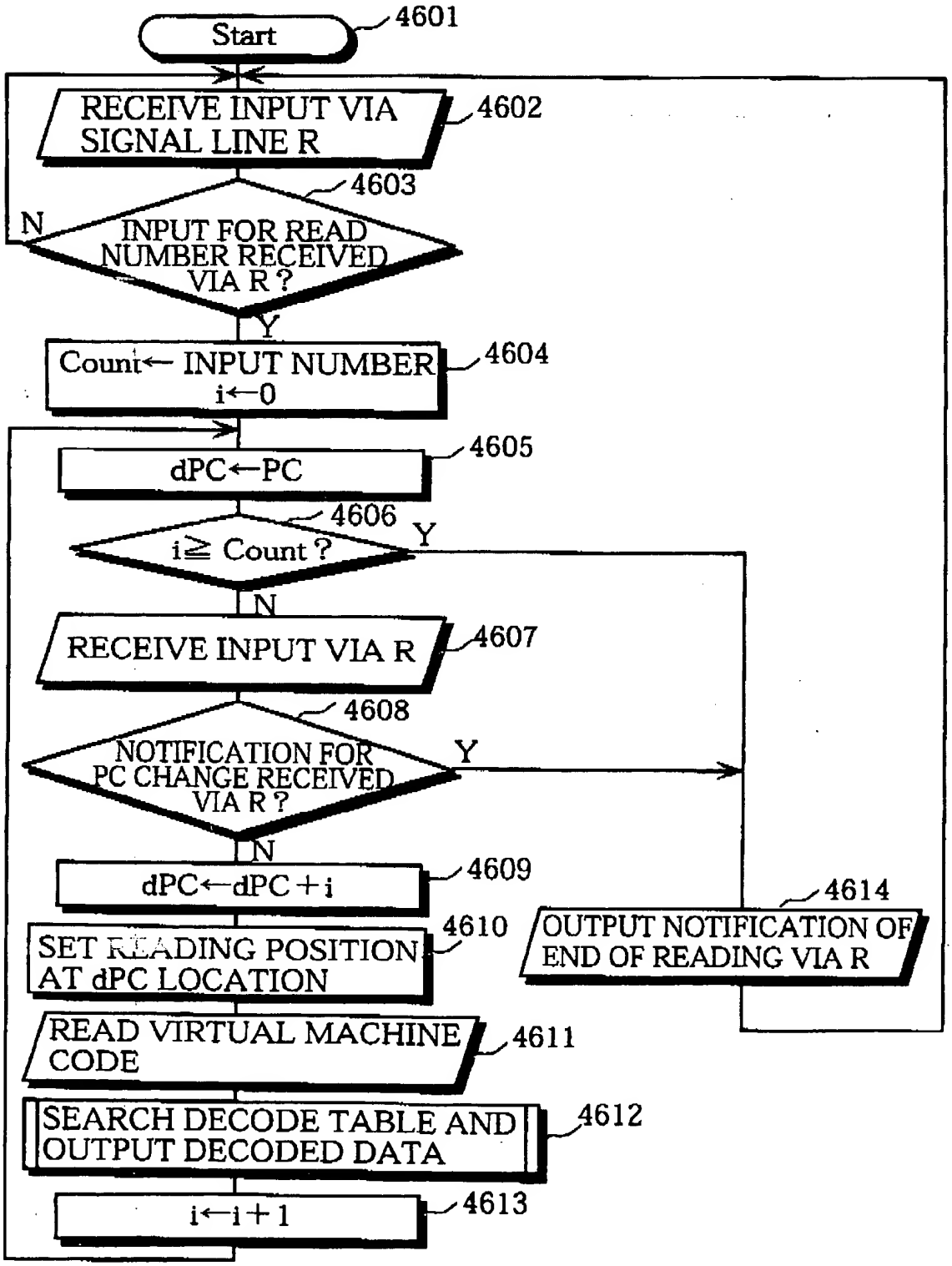

Start
4601
RECEIVE INPUT VIA SIGNAL LINE R
4602
4603
INPUT FOR READ NUMBER RECEIVED VIA R ?
N
Y
Count← INPUT NUMBER
i←0
4604
dPC←PC
4605
4606
i≧ Count ?
Y
N
RECEIVE INPUT VIA R
4607
4608
NOTIFICATION FOR PC CHANGE RECEIVED VIA R ?
Y
N
dPC←dPC+i
4609
SET READING POSITION AT dPC LOCATION
4610
READ VIRTUAL MACHINE CODE
4611
SEARCH DECODE TABLE AND OUTPUT DECODED DATA
4612
i←i+1
4613
OUTPUT NOTIFICATION OF END OF READING VIA R
4614

FIG. 9

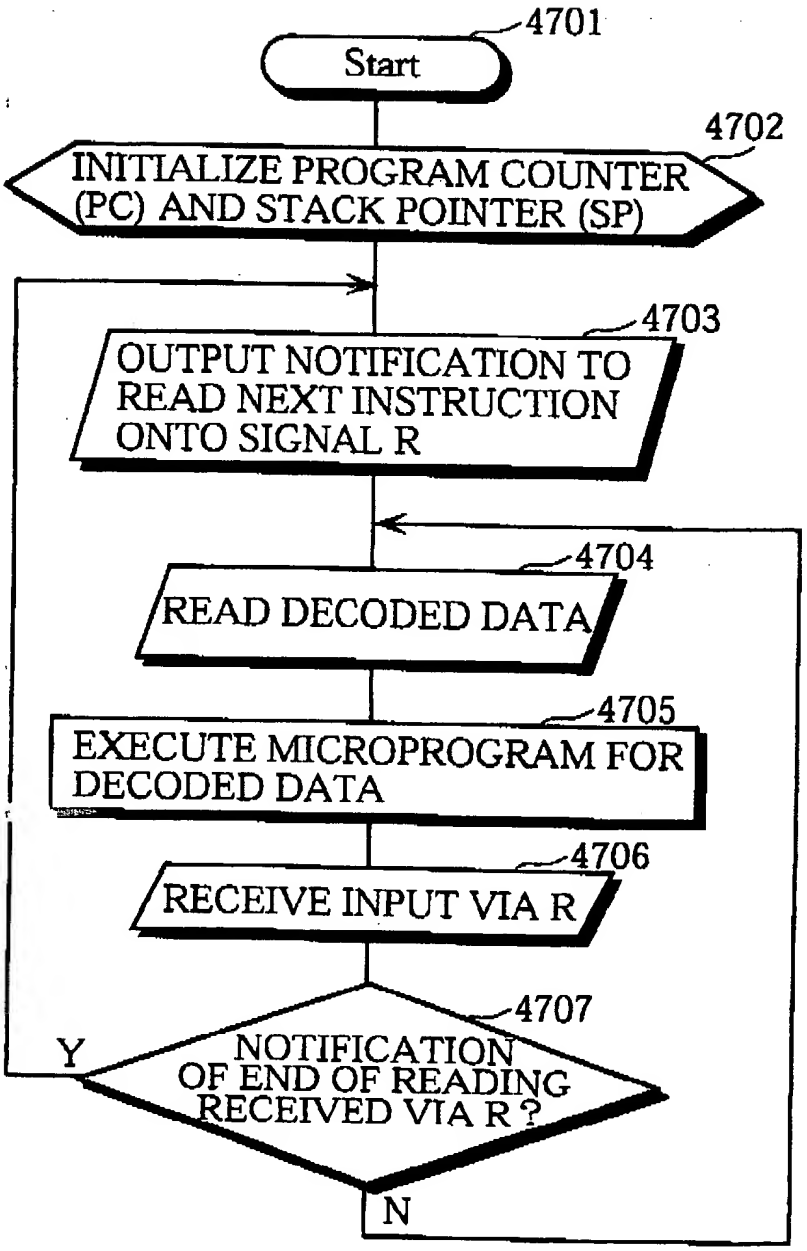

4701
Start
4702
INITIALIZE PROGRAM COUNTER (PC) AND STACK POINTER (SP)
4703
OUTPUT NOTIFICATION TO READ NEXT INSTRUCTION ONTO SIGNAL R
4704
READ DECODED DATA
4705
EXECUTE MICROPROGRAM FOR DECODED DATA
4706
RECEIVE INPUT VIA R
4707
NOTIFICATION OF END OF READING RECEIVED VIA R ?
Y
N

FIG. 10A

1:	Push
2:	2
3:	Push
4:	3
5:	Push
6:	4
7:	Add
8:	Mult
9:	Pop
10:	0

FIG. 10B

ARITHMETIC EXPRESSION : <DATA AREA #0>=2*(3+4)

FIG. 10C

1: <JUMP ADDRESS OF CODE TO PERFORM Push>
2: OPERAND "2"
3: <JUMP ADDRESS OF CODE TO PERFORM Push>
4: OPERAND "3"
5: <JUMP ADDRESS OF CODE TO PERFORM Push>
6: OPERAND "4"
7: <JUMP ADDRESS OF CODE TO PERFORM Add>
8: <JUMP ADDRESS OF CODE TO PERFORM Mult>
9: <JUMP ADDRESS OF CODE TO PERFORM Pop>
10: OPERAND "0"

FIG. 11A

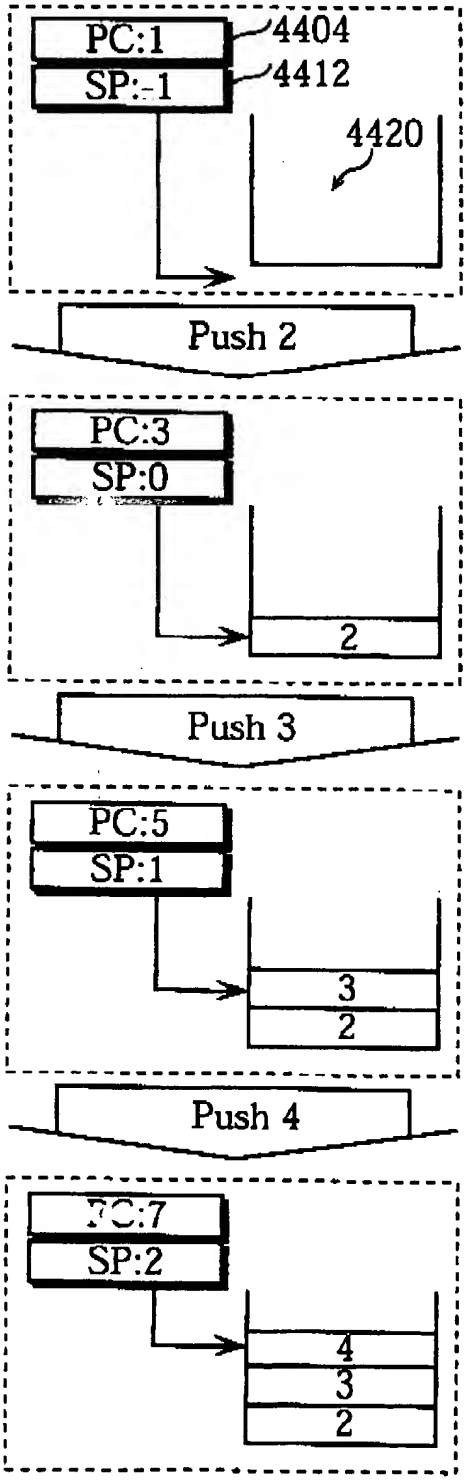


FIG. 11B

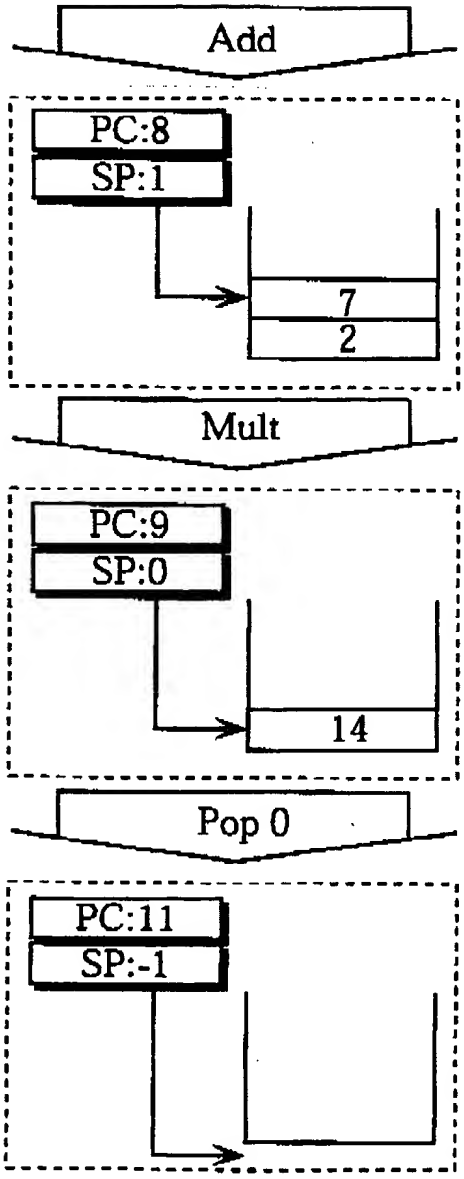


FIG. 12A

```
MICROPROGRAM FOR VIRTUAL MACHINE INSTRUCTION "Push"
1:Inc      r3        ; INCREMENT SP VALUE BY ONE
2:Store    [r3],r0   ; PLACE VALUE OF TOS REGISTER (#0)
                     ; INTO STACK
3:Load     r0,[r2]   ; EXTRACT OPERAND AND
                     ; PLACE IT ONTO TOS REGISTER
4:Inc      r2        ; INCREMENT PC OF VIRTUAL MACHINE BY ONE TO PREPARE
                     ; FOR READING NEXT INSTRUCTION
<MICROPROGRAM FOR JUMPING TO NEXT VIRTUAL MACHINE INSTRUCTION>
```

FIG. 12B

```
MICROPROGRAM FOR VIRTUAL MACHINE INSTRUCTION "Add"
1:Load     r1,[r3]   ; EXTRACT VALUE FROM STACK
                     ; PLACE IT ONTO REGISTER #1
2:Dec      r3        ; DECREMENT VALUE OF VIRTUAL MACHINE PC BY ONE
3:Add      r0,r0,r1  ; ADD VALUES OF REGISTERS #0 AND #1 AND
                     ; PLACE RESULT ONTO TOS REGISTER

<MICROPROGRAM FOR JUMPING TO NEXT VIRTUAL MACHINE INSTRUCTION>
```

FIG. 12C

```
MICROPROGRAM FOR VIRTUAL MACHINE INSTRUCTION "Mult"
1:Load     r1,[r3]   ; EXTRACT VALUE FROM STACK AND
                     ; PLACE IT ONTO REGISTER #1
2:Dec      r3        ; DECREMENT VALUE OF VIRTUAL MACHINE SP BY ONE
3:Mult     r0,r0,r1  ; MULTIPLY VALUES OF REGISTERS #0 AND #1 AND
                     ; PLACE RESULT ONTO TOS REGISTER

<MICROPROGRAM FOR JUMPING TO NEXT VIRTUAL MACHINE INSTRUCTION>
```

FIG. 12D

```
<MICROPROGRAM FOR JUMPING TO NEXT VIRTUAL MACHINE INSTRUCTION>
1:Load     r1,[r2]   ; READ VIRTUAL MACHINE INSTRUCTION (JUMP ADDRESS)
                     ; INDICATED BY PC INTO REGISTER #1
2:Inc      r2        ; INCREMENT VIRTUAL MACHINE PC BY ONE
3:Jmp      r1        ; JUMP UNCONDITIONALLY TO LOCATION
                       INDICATED BY REGISTER #1
```

FIG. 13A

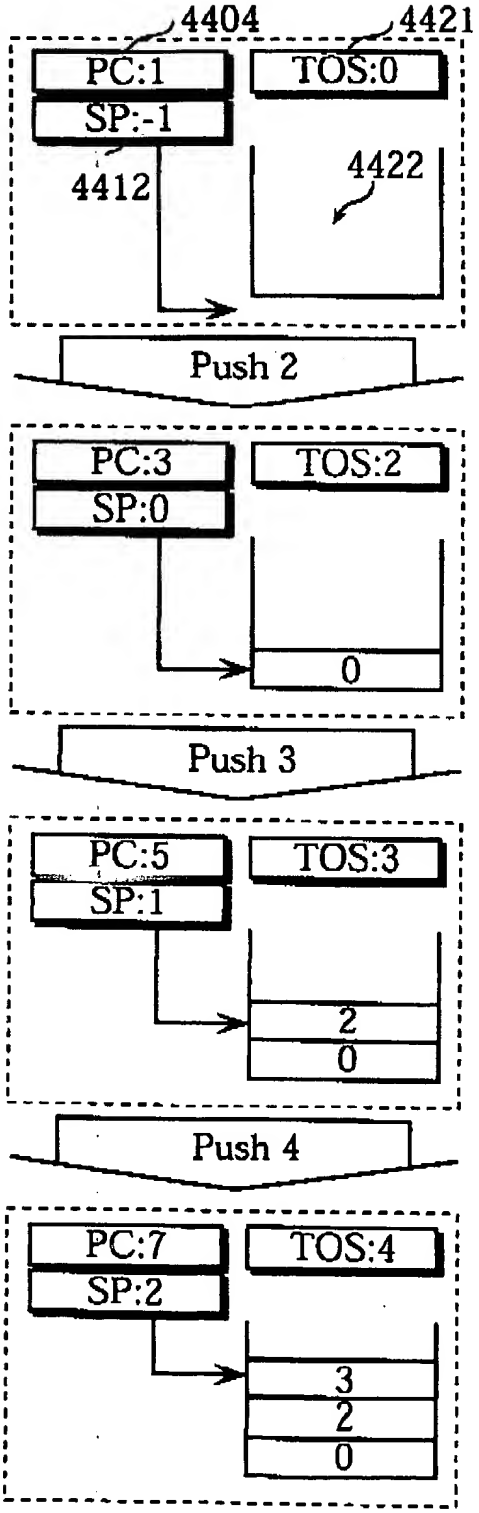


FIG. 13B

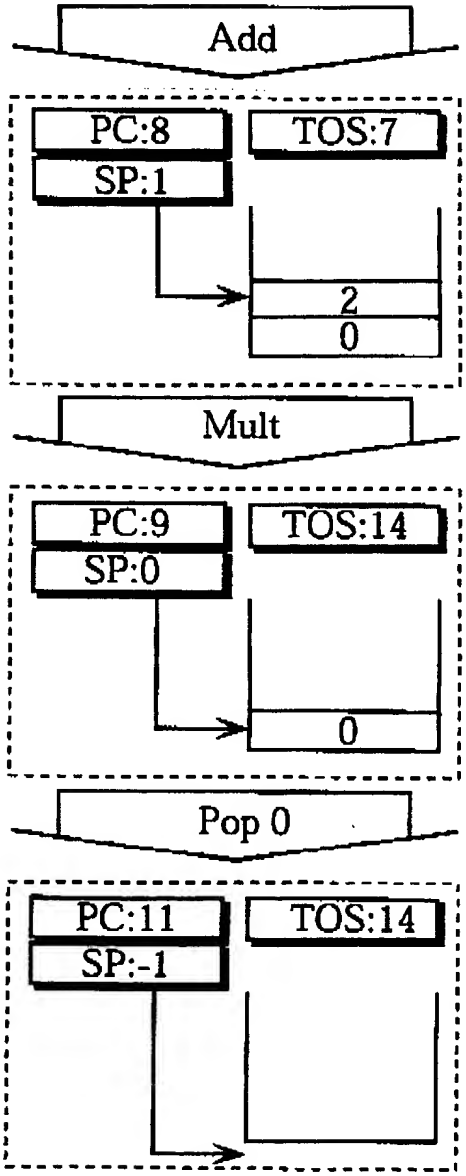


FIG. 14

STAGE NAME	NOTATION
INSTRUCTION FETCH	IF
INSTRUCTION DECODE AND REGISTER REFERENCE	RF
EXECUTION	ALU
MEMORY ACCESS	MEM
RESULT WRITING TO REGISTER	WB

FIG. 15

CLOCK	1	2	3	4	5	6	7	8
INSTRUCTION A	IF	RF	ALU	MEM	WB			
INSTRUCTION B		IF	RF	ALU	MEM	WB		
INSTRUCTION C			IF	RF	ALU	MEM	WB	
INSTRUCTION D				IF	RF	ALU	MEM	WB

FIG. 16

CLOCK	1	2	3	4	5	6	7	8
INSTRUCTION A1	IF	RF	ALU	MEM	WB			
INSTRUCTION A2	IF	RF	ALU	MEM	WB			
INSTRUCTION B1		IF	RF	ALU	MEM	WB		
INSTRUCTION B2		IF	RF	ALU	MEM	WB		
INSTRUCTION C1			IF	RF	ALU	MEM	WB	
INSTRUCTION C2			IF	RF	ALU	MEM	WB	
INSTRUCTION D1				IF	RF	ALU	MEM	WB
INSTRUCTION D2				IF	RF	ALU	MEM	WB

FIG. 17

CLOCK	1	2	3	4	5	6	7	8	9
INSTRUCTION A	IF	RF	ALU	MEM	WB				
INSTRUCTION B		IF	RF	•	ALU	MEM	WB		
INSTRUCTION C			IF	•	RF	ALU	MEM	WB	
INSTRUCTION D				•	IF	RF	ALU	MEM	WB

FIG. 18

CLOCK	1	2	3	4	5	6	7	8	9
INSTRUCTION A1	IF	RF	ALU	MEM	WB				
INSTRUCTION A2	IF	RF	ALU	MEM	WB				
INSTRUCTION B1		IF	RF	•	ALU	MEM	WB		
INSTRUCTION B2		IF	RF	ALU	MEM	WB			
INSTRUCTION C1			IF	RF	ALU	MEM	WB		
INSTRUCTION C2			IF	RF	•	ALU	MEM	WB	
INSTRUCTION D1				IF	RF	ALU	MEM	WB	
INSTRUCTION D2				IF	RF	•	ALU	MEM	WB

FIG. 19

CLOCK	1	2	3	4	5	6	7	8	9	10
INSTRUCTION A	IF	RF	ALU	MEM	WB					
INSTRUCTION B		IF	RF	•	•	ALU	MEM	WB		
INSTRUCTION C			IF	•	•	RF	ALU	MEM	WB	
INSTRUCTION D				•	•	IF	RF	ALU	MEM	WB

FIG. 20

CLOCK	1	2	3	4	5	6	7	8	9
INSTRUCTION A1	IF	RF	ALU	MEM	WB				
INSTRUCTION A2	IF	RF	ALU	MEM	WB				
INSTRUCTION B1		IF	RF	•	•	ALU	MEM	WB	
INSTRUCTION B2		IF	RF	ALU	MEM	WB			
INSTRUCTION C1			IF	RF	ALU	MEM	WB		
INSTRUCTION C2			IF	RF	ALU	MEM	WB		
INSTRUCTION D1				IF	RF	ALU	MEM	WB	
INSTRUCTION D2				IF	RF	•	ALU	MEM	WB

FIG. 21

CLOCK	1	2	3	4	5	6	7
INSTRUCTION A	IF	RF	ALU	MEM	WB		
INSTRUCTION B		IF	x				
INSTRUCTION C			IF	RF	ALU	MEM	WB

FIG. 22

CLOCK	1	2	3	4	5	6	7
INSTRUCTION A1	IF	RF	ALU	MEM	WB		
INSTRUCTION A2	IF	RF	ALU	MEM	WB		
INSTRUCTION B1		IF	x				
INSTRUCTION B2		IF	x				
INSTRUCTION C1			IF	RF	ALU	MEM	WB
INSTRUCTION C2			IF	RF	ALU	MEM	WB

FIG. 23

CLOCK	1	2	3	4	5	6	7	8	9	10	11
Load r1,[r2]	IF	RF	ALU	MEM	WB						
Inc r2		IF	RF	ALU	MEM	WB					
Jmp r1			IF	RF	ALU	MEM	WB				
				IF	x						
Load r1,[r3]					IF	RF	ALU	MEM	WB		
Dec r3						IF	RF	ALU	MEM	WB	
Mult r0,r0,r1							IF	RF	ALU	MEM	WB

FIG. 24

CLOCK	1	2	3	4	5	6	7	8	9	10	11
Load r1,[r2]	IF	RF	ALU	MEM	WB						
Inc r2	IF	RF	ALU	MEM	WB						
Jmp r1		IF	RF	•	ALU	MEM	WB				
		IF	RF	ALU	x						
			IF	RF	x						
			IF	RF	x						
				IF	x						
				IF	x						
Load r1,[r3]					IF	RF	ALU	MEM	WB		
Dec r3					IF	RF	ALU	MEM	WB		
Mult r0,r0,r1						IF	RF	•	ALU	MEM	WB

6660n0 E9288260

FIG. 25

CLOCK	1	2	3	4	5	6	7	8	9	10	11	12	13
Load r1,[r2]	IF	RF	ALU	MEM	WB								
Inc r2		IF	RF	ALU	MEM	WB							
Jmp r1			IF	RF	•	ALU	MEM	WB					
				IF	•	x							
Load r1,[r3]					•	IF	RF	ALU	MEM	WB			
Dec r3							IF	RF	ALU	MEM	WB		
Mult r0,r0,r1								IF	RF	•	ALU	MEM	WB

6680h0 E9288260

FIG. 26

CLOCK	1	2	3	4	5	6	7	8	9	10	11	12	13
Load r1,[r2]	IF	RF	ALU	MEM	WB								
Inc r2	IF	RF	ALU	MEM	WB								
Jmp r1		IF	RF	•	•	ALU	MEM	WB					
		IF	RF	ALU	MEM	x							
			IF	RF	ALU	x							
			IF	RF	ALU	x							
				IF	RF	x							
				IF	RF	x							
					IF	x							
					IF	x							
Load r1,[r3]						IF	RF	ALU	MEM	WB			
Dec r3						IF	RF	ALU	MEM	WB			
Mult r0,r0,r1							IF	RF	•	•	ALU	MEM	WB

FIG. 27

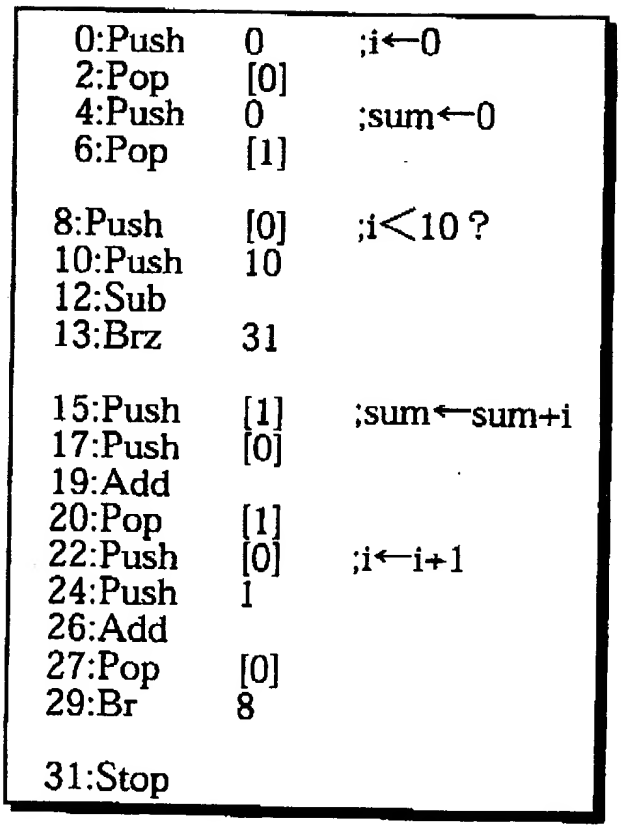
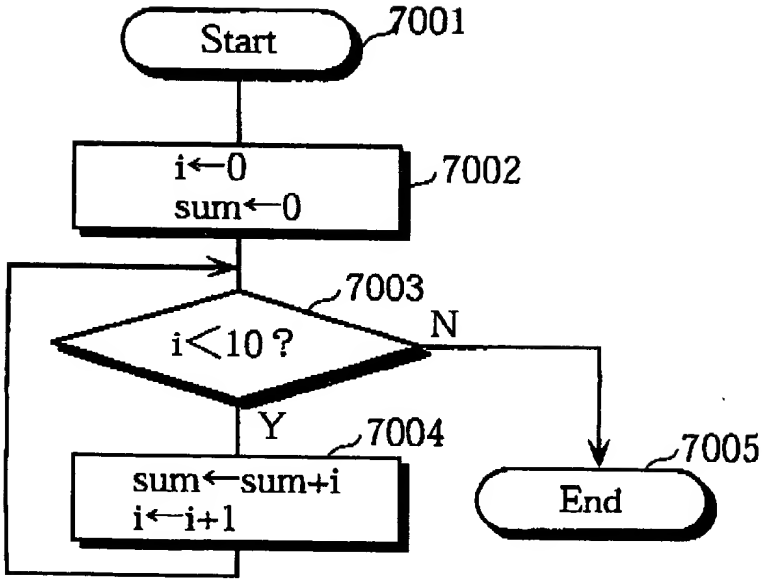

0:Push 0 ;i←0
2:Pop [0]
4:Push 0 ;sum←0
6:Pop [1]
8:Push [0] ;i<10 ?
10:Push 10
12:Sub
13:Brz 31
15:Push [1] ;sum←sum+i
17:Push [0]
19:Add
20:Pop [1]
22:Push [0] ;i←i+1
24:Push 1
26:Add
27:Pop [0]
29:Br 8
31:Stop

FIG. 28



6680n0 E9288260

FIG. 29

VIRTUAL MACHINE INSTRUCTION	SIZE OF REAL MACHINE CODE TEMPLATE	REAL MACHINE CODE TEMPLATE	NUMBER OF OPERANDS
:	:	:	:
Push	4 WORDS	<CODE TO PERFORM Push>	1
Pop	5 WORDS	<CODE TO PERFORM Pop>	1
Add	3 WORDS	<CODE TO PERFORM Add>	0
Sub	3 WORDS	<CODE TO PERFORM Sub>	0
Mult	3 WORDS	<CODE TO PERFORM Mult>	0
Push[]	5 WORDS	<CODE TO PERFORM Push[]>	1
Br	3 WORDS	<CODE TO PERFORM Br>	1
Brz	5 WORDS	<CODE TO PERFORM Brz>	1
Stop	2 WORDS	<CODE TO PERFORM Stop>	0
:	:	:	:

FIG. 30

VIRTUAL MACHINE CODE ADDRESS	VIRTUAL MACHINE CODE	REAL MACHINE CODE SIZE	CORRESPONDING REAL MACHINE CODE ADDRESS
0	Push 0	4	0-3
2	Pop [0]	5	4-8
4	Push 0	4	9-12
6	Pop [1]	5	13-17
8	Push [0]	5	18-22
10	Push 10	4	23-26
12	Sub	3	27-29
13	Brz 31	5	30-34
15	Push [1]	5	35-39
17	Push [0]	5	40-44
19	Add	3	45-47
20	Pop [1]	5	48-52
22	Push [0]	5	53-57
24	Push 1	4	58-61
26	Add	3	62-64
27	Pop [0]	5	65-69
29	Br 8	3	70-72
31	Stop	2	73-75

FIG. 31A

9300a 9300b

BIT SEQUENCE	MEANING
0	"a"
10	"b"
110	"c"
111	"d"

FIG. 31B

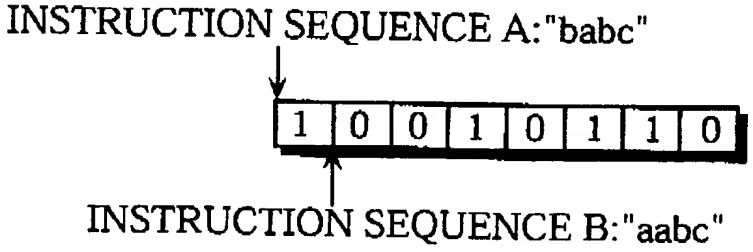


FIG. 32

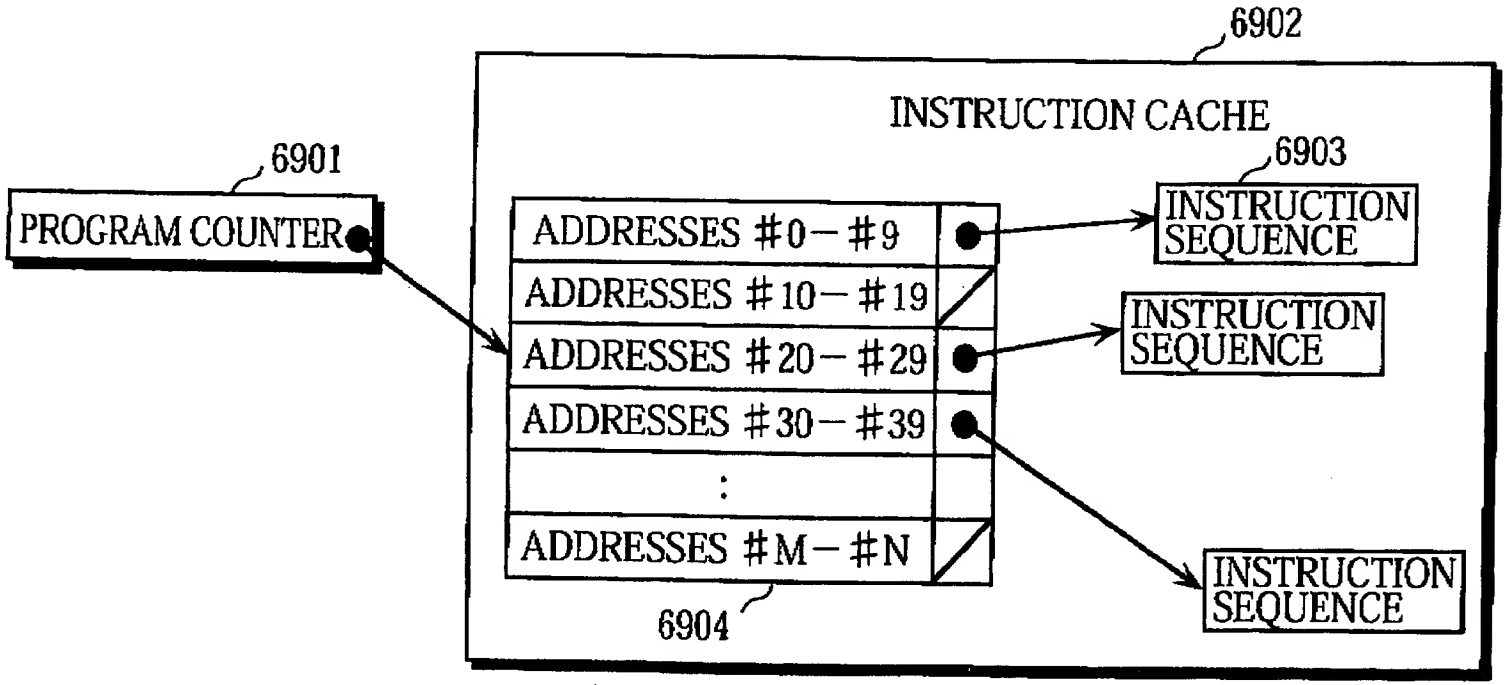

6902
INSTRUCTION CACHE
6901
PROGRAM COUNTER
6903
ADDRESSES #0－#9
ADDRESSES #10－#19
ADDRESSES #20－#29
ADDRESSES #30－#39
:
ADDRESSES #M－#N
INSTRUCTION SEQUENCE
INSTRUCTION SEQUENCE
INSTRUCTION SEQUENCE
6904

FIG. 33

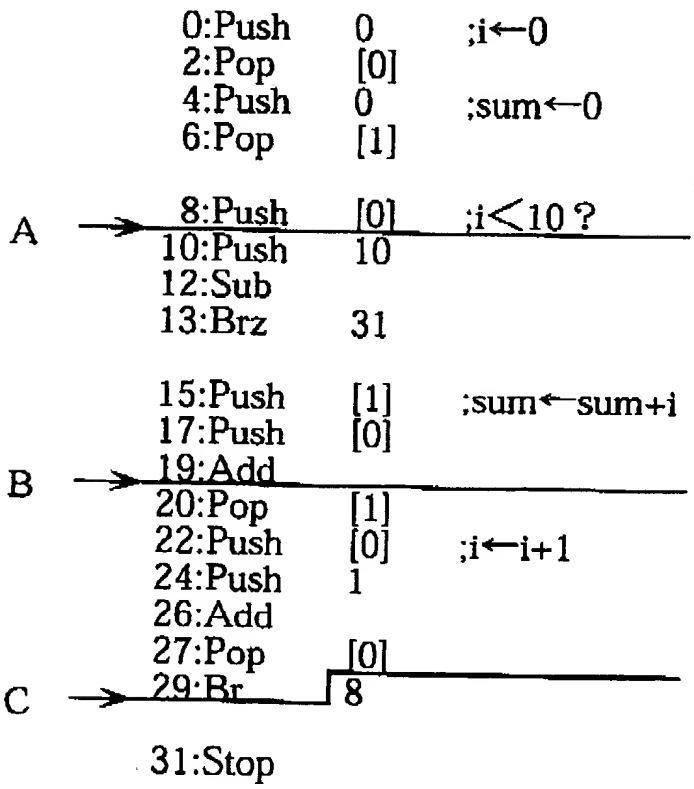

0:Push 0 ;i←0
2:Pop [0]
4:Push 0 ;sum←0
6:Pop [1]
A
8:Push [0] ;i<10?
10:Push 10
12:Sub
13:Brz 31
15:Push [1] ;sum←sum+i
17:Push [0]
B
19:Add
20:Pop [1]
22:Push [0] ;i←i+1
24:Push 1
26:Add
27:Pop [0]
C
29:Br 8
31:Stop

FIG. 34

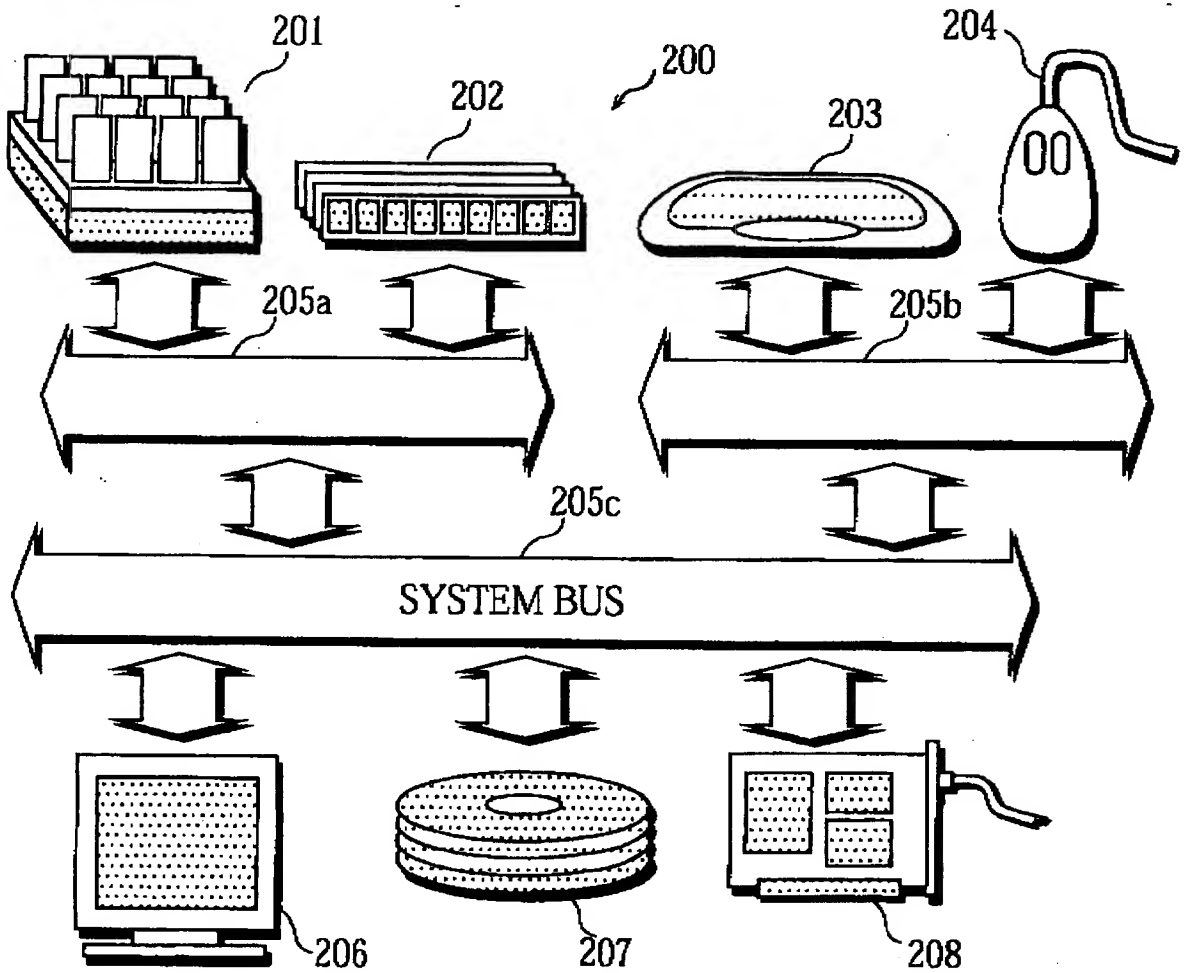


FIG. 35

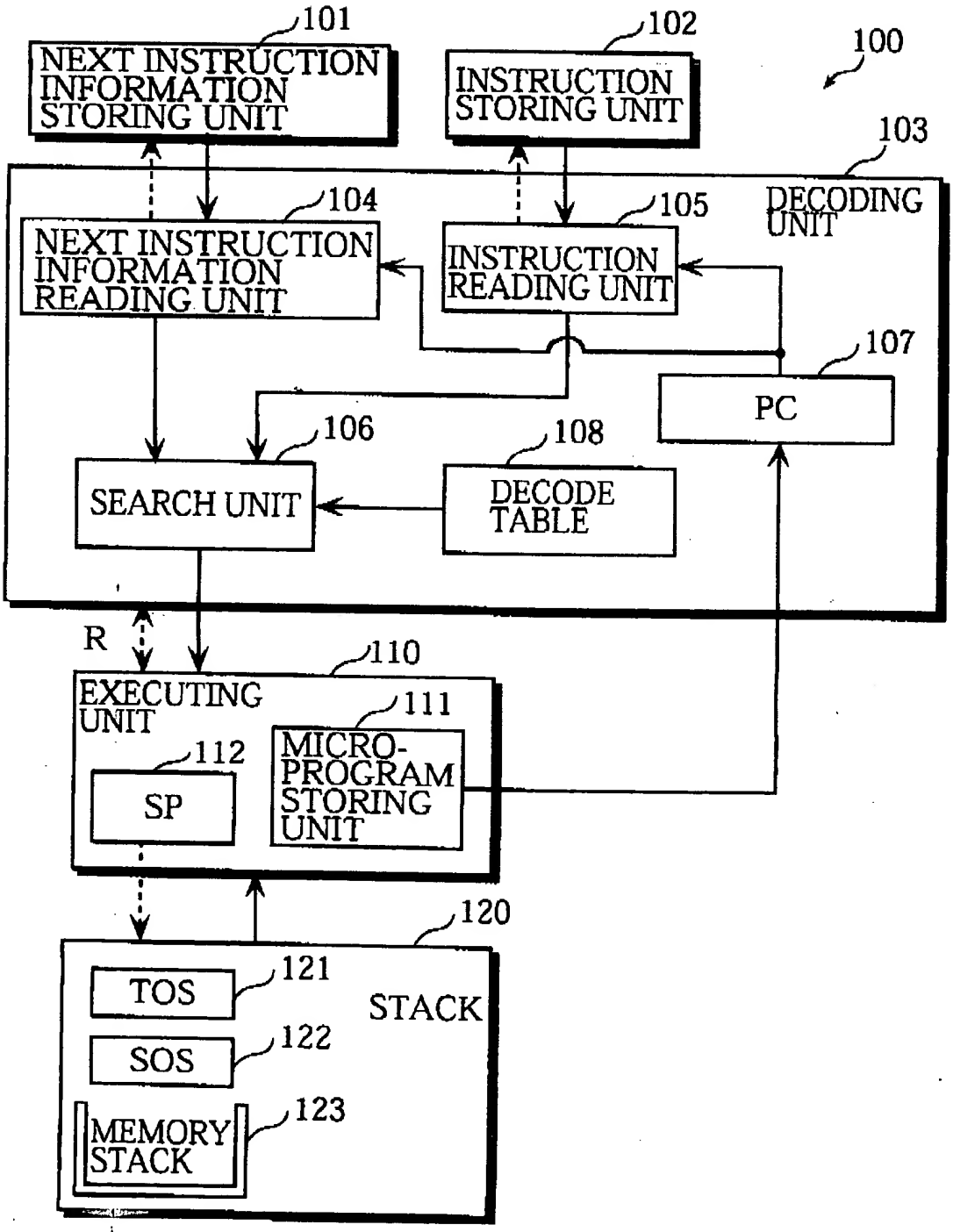

101
NEXT INSTRUCTION INFORMATION STORING UNIT
102
INSTRUCTION STORING UNIT
100
103
DECODING UNIT
104
NEXT INSTRUCTION INFORMATION READING UNIT
105
INSTRUCTION READING UNIT
107
PC
106
SEARCH UNIT
108
DECODE TABLE
R
110
EXECUTING UNIT
111
MICRO-PROGRAM STORING UNIT
112
SP
120
TOS
121
STACK
SOS
122
MEMORY STACK
123

FIG. 36A

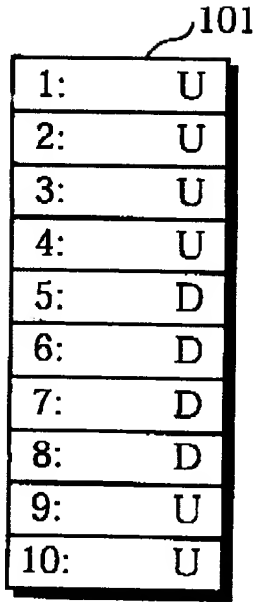
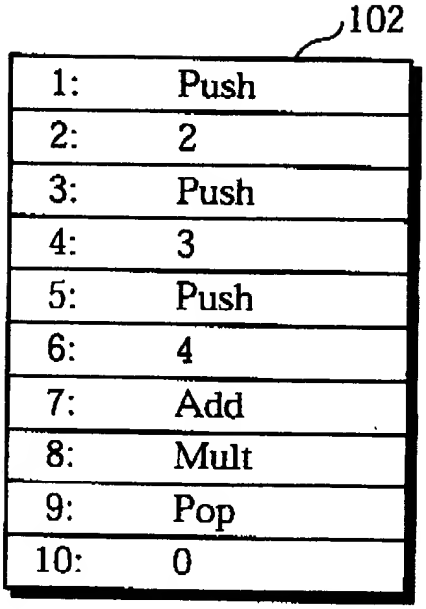


FIG. 36B



08801n0 EP2288260

FIG. 37

108

108a OPCODE	108b NEXT INSTRUCTION INFORMATION	108c JUMP ADDRESS	108d NUMBER OF OPERANDS
:	:	:	:
Push	U	<JUMP ADDRESS OF CODE TO PERFORM Push ASSIGNED "U">	1
Push	D	<JUMP ADDRESS OF CODE TO PERFORM Push ASSIGNED "D">	1
Pop	U	<JUMP ADDRESS OF CODE TO PERFORM Pop ASSIGNED "U">	1
Pop	D	<JUMP ADDRESS OF CODE TO PERFORM Pop ASSIGNED "D">	1
Add	U	<JUMP ADDRESS OF CODE TO PERFORM Add ASSIGNED "U">	0
Add	D	<JUMP ADDRESS OF CODE TO PERFORM Add ASSIGNED "D">	0
Sub	U	<JUMP ADDRESS OF CODE TO PERFORM Sub ASSIGNED "U">	0
Sub	D	<JUMP ADDRESS OF CODE TO PERFORM Sub ASSIGNED "D">	0
Inc	U	<JUMP ADDRESS OF CODE TO PERFORM Inc ASSIGNED "U">	0
Inc	D	<JUMP ADDRESS OF CODE TO PERFORM Inc ASSIGNED "D">	0
:	:	:	:

FIG. 38A

MICROPROGRAM FOR VIRTUAL MACHINE INSTRUCTION "Push" WITH "U"		
1:Load	r4,r0	;COPY VALUE OF TOS REGISTER (#0) INTO SOS REGISTER (#4)
2:Load	r0,[r2]	;READ OPERAND INTO TOS REGISTER
3:Inc	r2	;INCREMENT VIRTUAL MACHINE PC BY ONE TO PREPARE FOR READING NEXT INSTRUCTION
4:Inc	r3	;INCREMENT VIRTUAL MACHINE SP BY ONE
5:Store	[r3],r4	;PLACE SOS REGISTER VALUE INTO STACK
6:Load	r1,[r2]	;READ VIRTUAL MACHINE INSTRUCTION (JUMP ADDRESS) INDICATED BY PC INTO REGISTER #1
7:Inc	r2	;INCREMENT VIRTUAL MACHINE PC BY ONE
8:Jmp	r1	;JUMP UNCONDITIONALLY TO LOCATION INDICATED BY REGISTER #1

FIG. 38B

MICROPROGRAM FOR VIRTUAL MACHINE INSTRUCTION "Push" WITH "D"		
1:Load	r4,r0	;COPY VALUE OF TOS REGISTER (#0) INTO SOS REGISTER (#4)
2:Load	r0,[r2]	;READ OPERAND INTO TOS REGISTER
3:Inc	r2	;INCREMENT VIRTUAL MACHINE PC BY ONE TO PREPARE FOR READING NEXT INSTRUCTION
6:Load	r1,[r2]	;READ VIRTUAL MACHINE INSTRUCTION (JUMP ADDRESS) INDICATED BY PC INTO REGISTER #1
7:Inc	r2	;INCREMENT VIRTUAL MACHINE PC BY ONE
8:Jmp	r1	;JUMP UNCONDITIONALLY TO LOCATION INDICATED BY REGISTER #1

0680n0 E9288260

FIG. 39A

MICROPROGRAM FOR VIRTUAL MACHINE INSTRUCTION "Add" WITH "U"
1:Add r0,r0,r4 ;ADD VALUES OF TOS REGISTER AND SOS REGISTER, AND PLACE RESULT INTO TOS REGISTER <MICROPROGRAM FOR VIRTUAL MACHINE JUMP CODE WITH "U">

FIG. 39B

MICROPROGRAM FOR VIRTUAL MACHINE INSTRUCTION "Add" WITH "D"
1:Add r0,r0,r4 ;ADD VALUES OF TOS REGISTER AND SOS REGISTER, AND PLACE RESULT INTO TOS REGISTER <MICROPROGRAM FOR VIRTUAL MACHINE JUMP CODE WITH "D">

6680nO E9288260

FIG. 40A

MICROPROGRAM FOR VIRTUAL MACHINE INSTRUCTION "Mult" WITH "U"
1:Mult r0,r0,r4 ;MULTIPLY VALUES OF TOS REGISTER AND SOS REGISTER, AND PLACE RESULT INTO TOS REGISTER <MICROPROGRAM FOR VIRTUAL MACHINE JUMP CODE WITH "U">

FIG. 40B

MICROPROGRAM FOR VIRTUAL MACHINE INSTRUCTION "Mult" WITH "D"
1:Mult r0,r0,r4 ;MULTIPLY VALUES OF TOS REGISTER AND SOS REGISTER, AND PLACE RESULT INTO TOS REGISTER <MICROPROGRAM FOR VIRTUAL MACHINE JUMP CODE WITH "D">

FIG. 41A

```
MICROPROGRAM FOR VIRTUAL MACHINE JUMP CODE WITH "U"
1:Load  r1,[r2]  ;READ VIRTUAL MACHINE INSTRUCTION
                  (JUMP ADDRESS) INDICATED BY VIRTUAL
                  MACHINE PC INTO REGISTER #1
2:Inc   r2       ;INCREMENT VALUE OF VIRTUAL MACHINE PC BY ONE
3:Jmp   r1       ;JUMP UNCONDITIONALLY TO LOCATION
                  INDICATED BY REGISTER #1
```

FIG. 41B

```
MICROPROGRAM FOR VIRTUAL MACHINE JUMP CODE WITH "D"
1:Load  r1,[r2]  ;READ VIRTUAL MACHINE INSTRUCTION
                  (JUMP ADDRESS) INDICATED BY VIRTUAL
                  MACHINE PC INTO REGISTER #1
2:Load  r4,[r3]  ;READ VALUE FROM STACK AND COPY IT
                  INTO SOS REGISTER
3:Inc   r2       ;INCREMENT VALUE OF VIRTUAL MACHINE PC BY ONE
4:Dec   r3       ;DECREMENT VALUE OF VIRTUAL MACHINE SP BY ONE
5:Jmp   r1       ;JUMP UNCONDITIONALLY TO LOCATION
                  INDICATED BY REGISTER #1
```

6680nD E9286260

FIG. 42

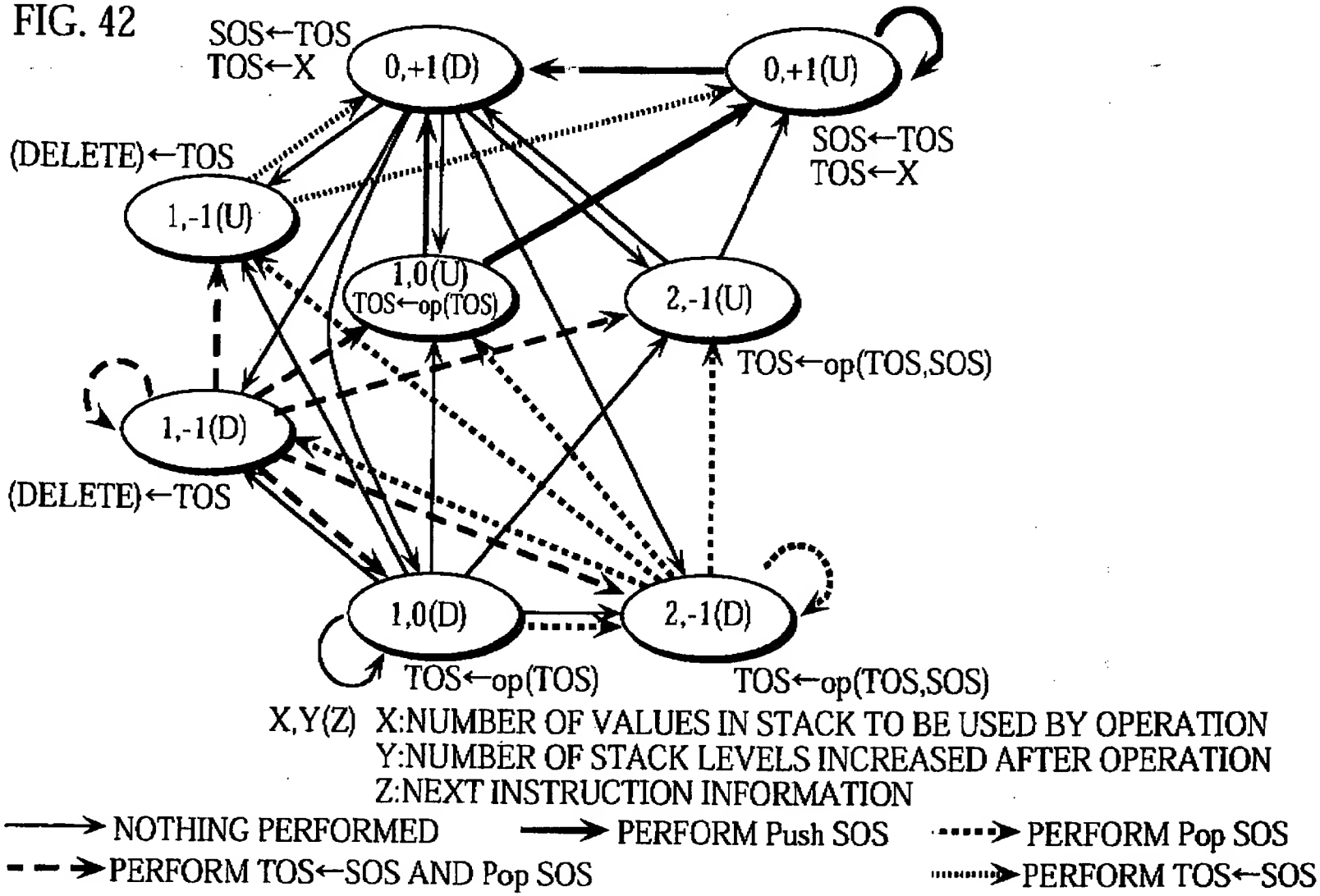


FIG. 43

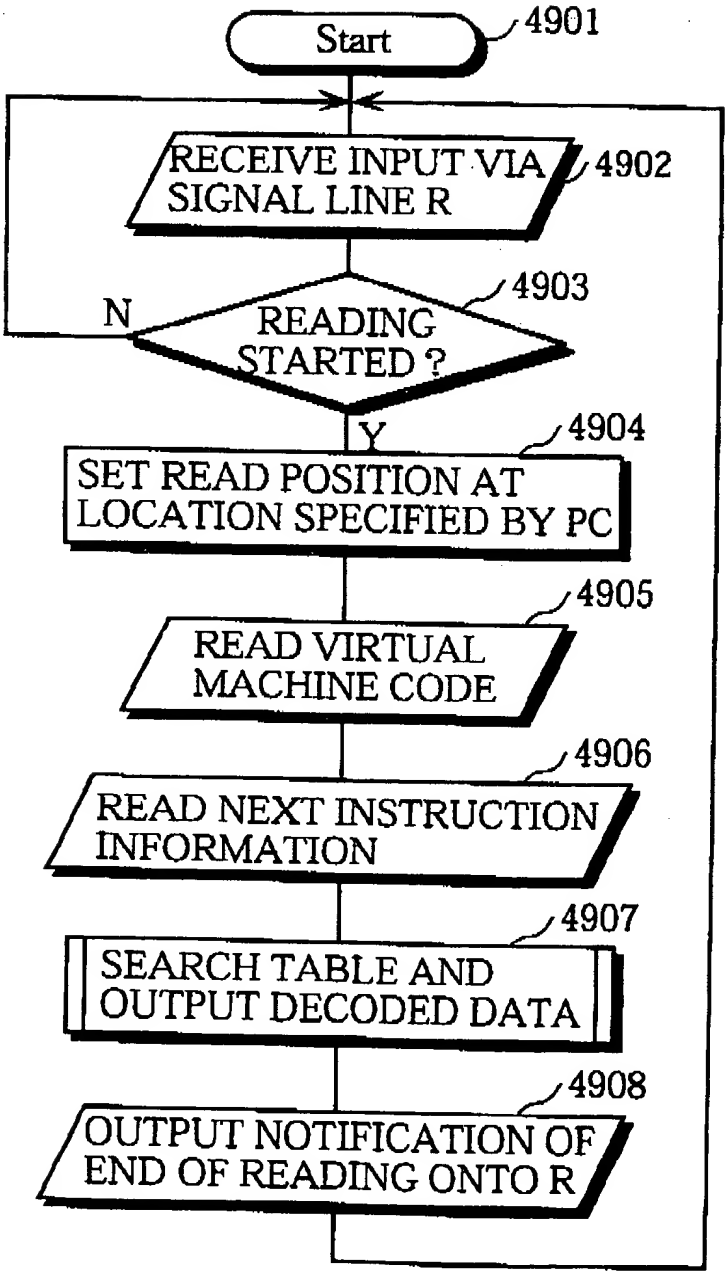

Start
4901
RECEIVE INPUT VIA SIGNAL LINE R
4902
N
READING STARTED ?
4903
Y
SET READ POSITION AT LOCATION SPECIFIED BY PC
4904
READ VIRTUAL MACHINE CODE
4905
READ NEXT INSTRUCTION INFORMATION
4906
SEARCH TABLE AND OUTPUT DECODED DATA
4907
OUTPUT NOTIFICATION OF END OF READING ONTO R
4908

FIG. 44

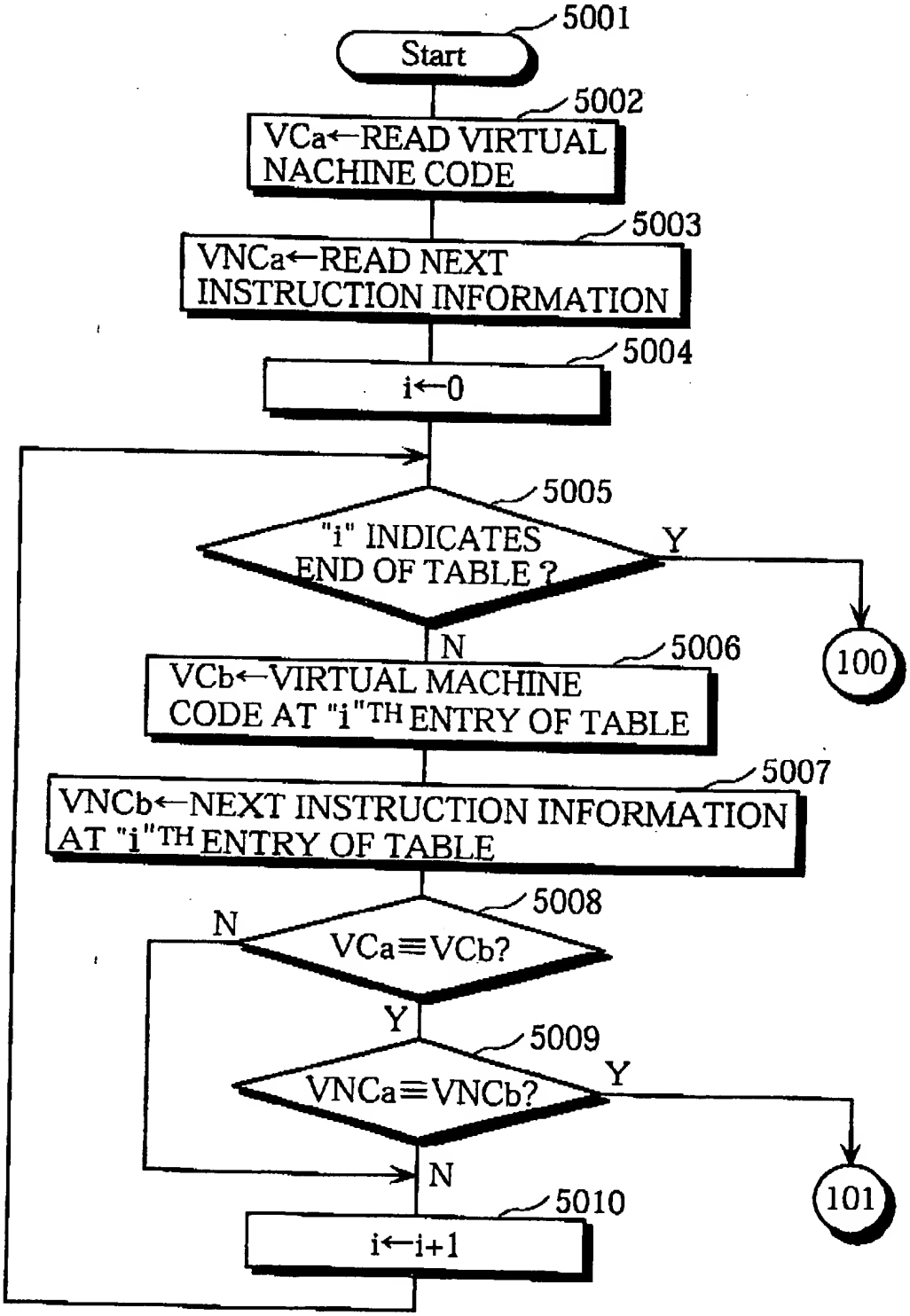

5001
Start
5002
VCa←READ VIRTUAL NACHINE CODE
5003
VNCa←READ NEXT INSTRUCTION INFORMATION
5004
i←0
5005
"i" INDICATES END OF TABLE ?
Y
N
100
5006
VCb←VIRTUAL MACHINE CODE AT "i"TH ENTRY OF TABLE
5007
VNCb←NEXT INSTRUCTION INFORMATION AT "i"TH ENTRY OF TABLE
5008
N
VCa≡VCb?
Y
5009
VNCa≡VNCb?
Y
N
101
5010
i←i+1

FIG. 45

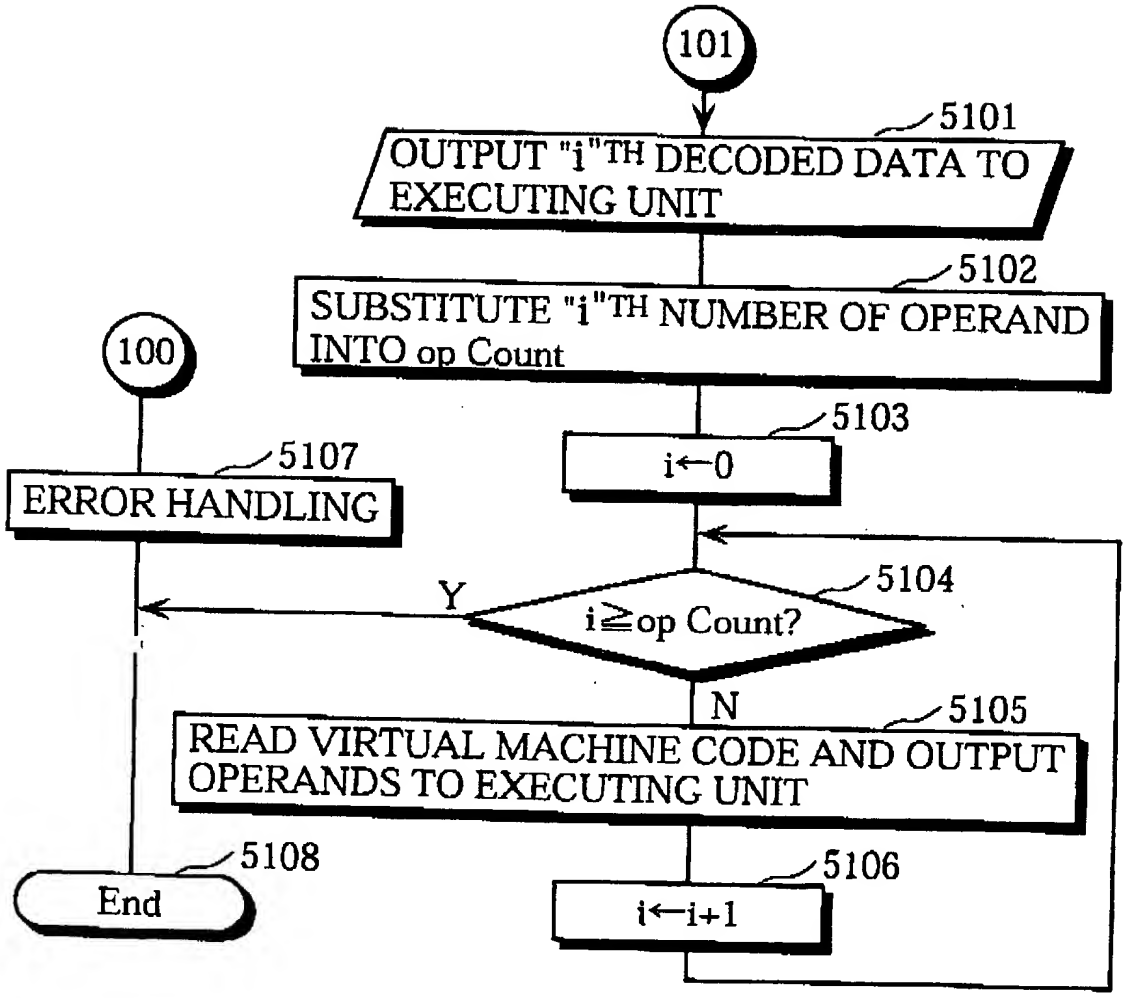

101
5101
OUTPUT "i"TH DECODED DATA TO EXECUTING UNIT
5102
SUBSTITUTE "i"TH NUMBER OF OPERAND INTO op Count
100
5103
i←0
5107
ERROR HANDLING
5104
Y
i≧op Count?
N
5105
READ VIRTUAL MACHINE CODE AND OUTPUT OPERANDS TO EXECUTING UNIT
5108
End
5106
i←i+1

FIG. 46

1:<JUMP ADDRESS OF CODE TO PERFORM Push WITH "U">
2:OPERAND "2"
3:<JUMP ADDRESS OF CODE TO PERFORM Push WITH "U">
4:OPERAND "3"
5:<JUMP ADDRESS OF CODE TO PERFORM Push WITH "D">
6:OPERAND "4"
7:<JUMP ADDRESS OF CODE TO PERFORM Add WITH "D">
8:<JUMP ADDRESS OF CODE TO PERFORM Mult WITH "D">
9:<JUMP ADDRESS OF CODE TO PERFORM Pop WITH "U">
10:OPERAND "0"

FIG. 47A

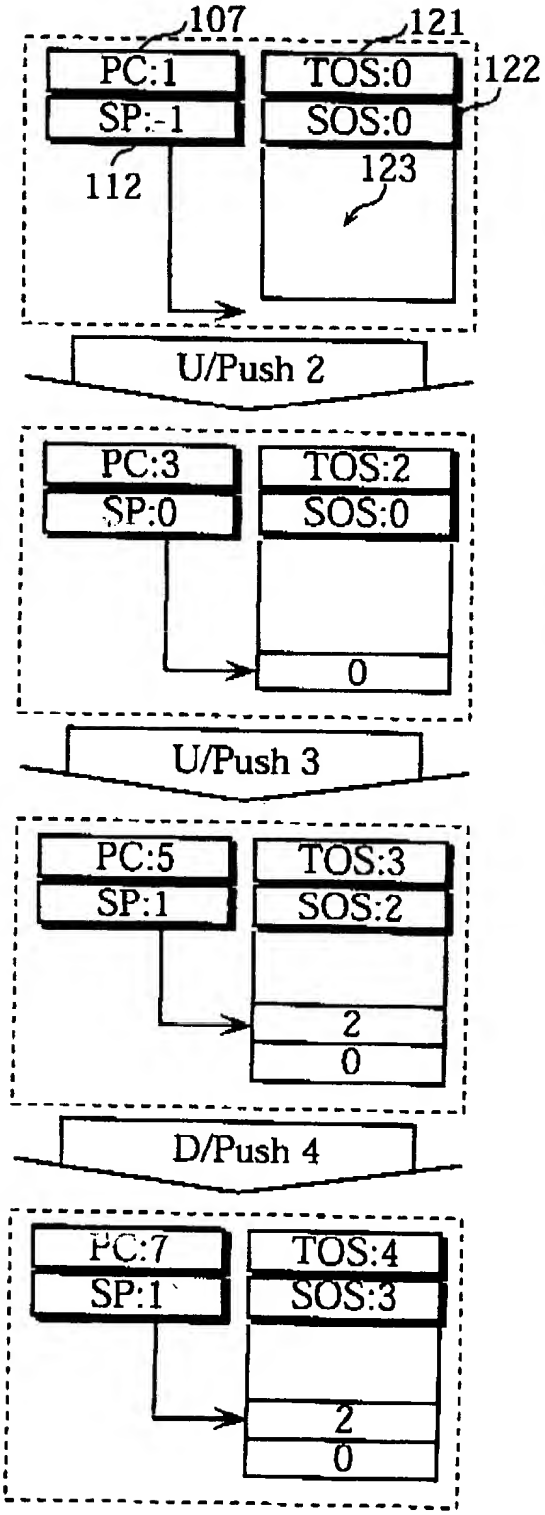


FIG. 47B

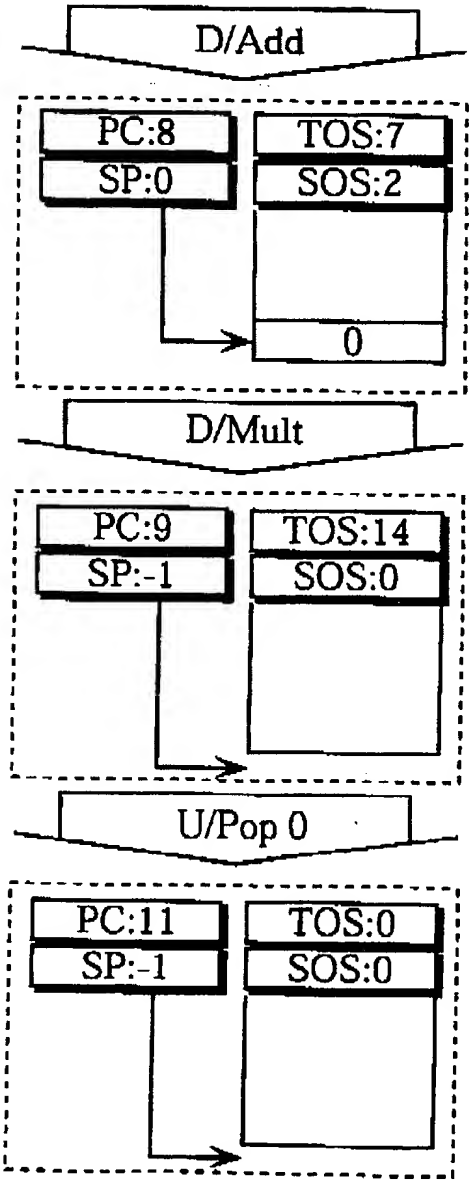


FIG. 48

CLOCK	1	2	3	4	5	6	7	8	9	10	11
Load r1,[r2]	IF	RF	ALU	MEM	WB						
Load r4,[r3]		IF	RF	ALU	MEM	WB					
Inc r2			IF	RF	ALU	MEM	WB				
Dec r3				IF	RF	ALU	MEM	WB			
Jmp r1					IF	RF	ALU	MEM	WB		
						IF	x				
Mult r0,r0,r4							IF	RF	ALU	MEM	WB

FIG. 49

CLOCK	1	2	3	4	5	6	7	8	9
Load r1,[r2]	IF	RF	ALU	MEM	WB				
Load r4,[r3]	IF	RF	ALU	MEM	WB				
Inc r2		IF	RF	ALU	MEM	WB			
Dec r3		IF	RF	ALU	MEM	WB			
Jmp r1			IF	RF	ALU	MEM	WB		
			IF	RF	x				
				IF	x				
				IF	x				
Mult r0,r0,r4					IF	RF	ALU	MEM	WB

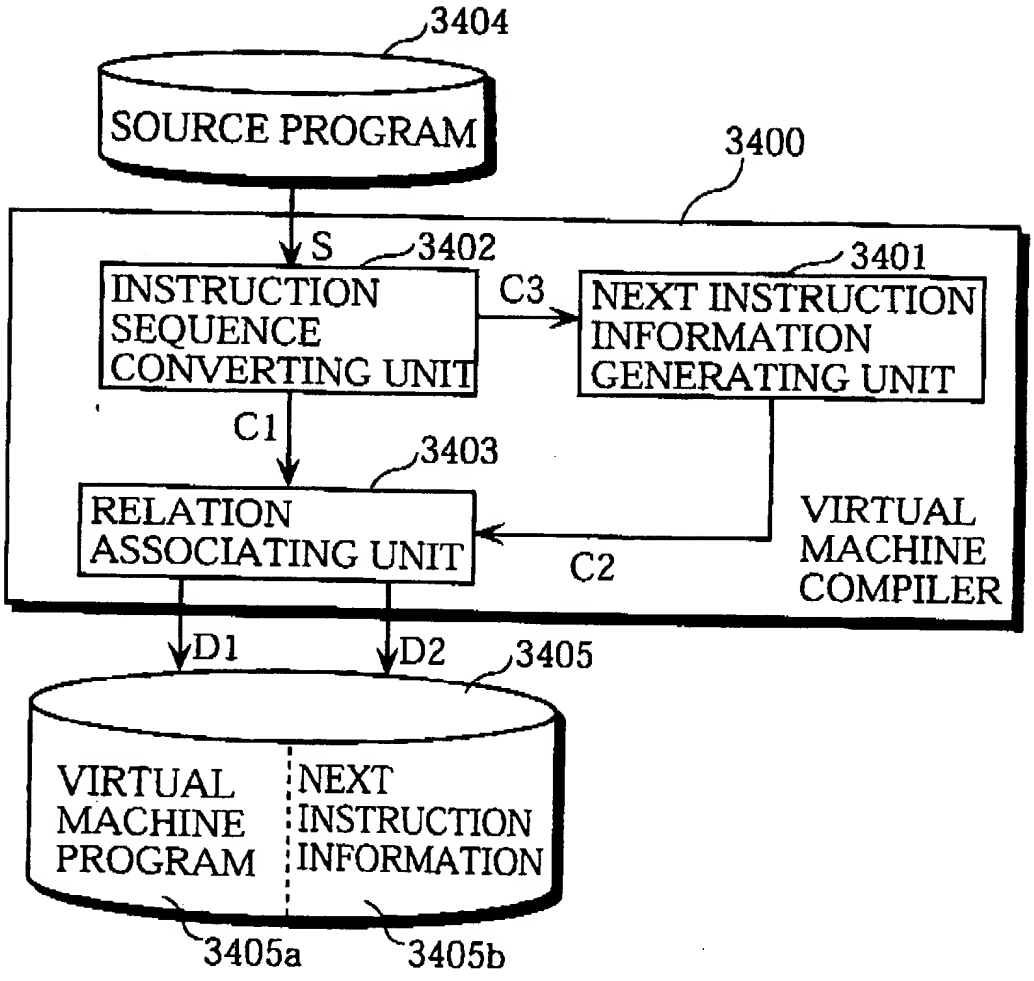
FIG. 50

CLOCK	1	2	3	4	5	6	7	8	9	10	11
Load r1,[r2]	IF	RF	ALU	MEM	WB						
Load r4,[r3]		IF	RF	ALU	MEM	WB					
Inc r2			IF	RF	ALU	MEM	WB				
Dec r3				IF	RF	ALU	MEM	WB			
Jmp r1					IF	RF	ALU	MEM	WB		
						IF	x				
Mult r0,r0,r4							IF	RF	ALU	MEM	WB

FIG. 51

CLOCK	1	2	3	4	5	6	7	8	9	10
Load r1,[r2]	IF	RF	ALU	MEM	WB					
Load r4,[r3]	IF	RF	ALU	MEM	WB					
Inc r2		IF	RF	ALU	MEM	WB				
Dec r3		IF	RF	ALU	MEM	WB				
Jmp r1			IF	RF	·	ALU	MEM	WB		
			IF	RF	ALU	x				
				IF	RF	x				
				IF	RF	x				
					IF	x				
					IF	x				
Mult r0,r0,r4						IF	RF	ALU	MEM	WB

FIG. 52


3404
SOURCE PROGRAM
3400
S
3402
INSTRUCTION SEQUENCE CONVERTING UNIT
C3
3401
NEXT INSTRUCTION INFORMATION GENERATING UNIT
C1
3403
RELATION ASSOCIATING UNIT
C2
VIRTUAL MACHINE COMPILER
D1
D2
3405
VIRTUAL MACHINE PROGRAM
NEXT INSTRUCTION INFORMATION
3405a
3405b

6680no E928826O

FIG. 53

DATA CONSTRUCTION OF
INSTRUCTION SEQUENCE:"x=(1+2)*(3+4)"

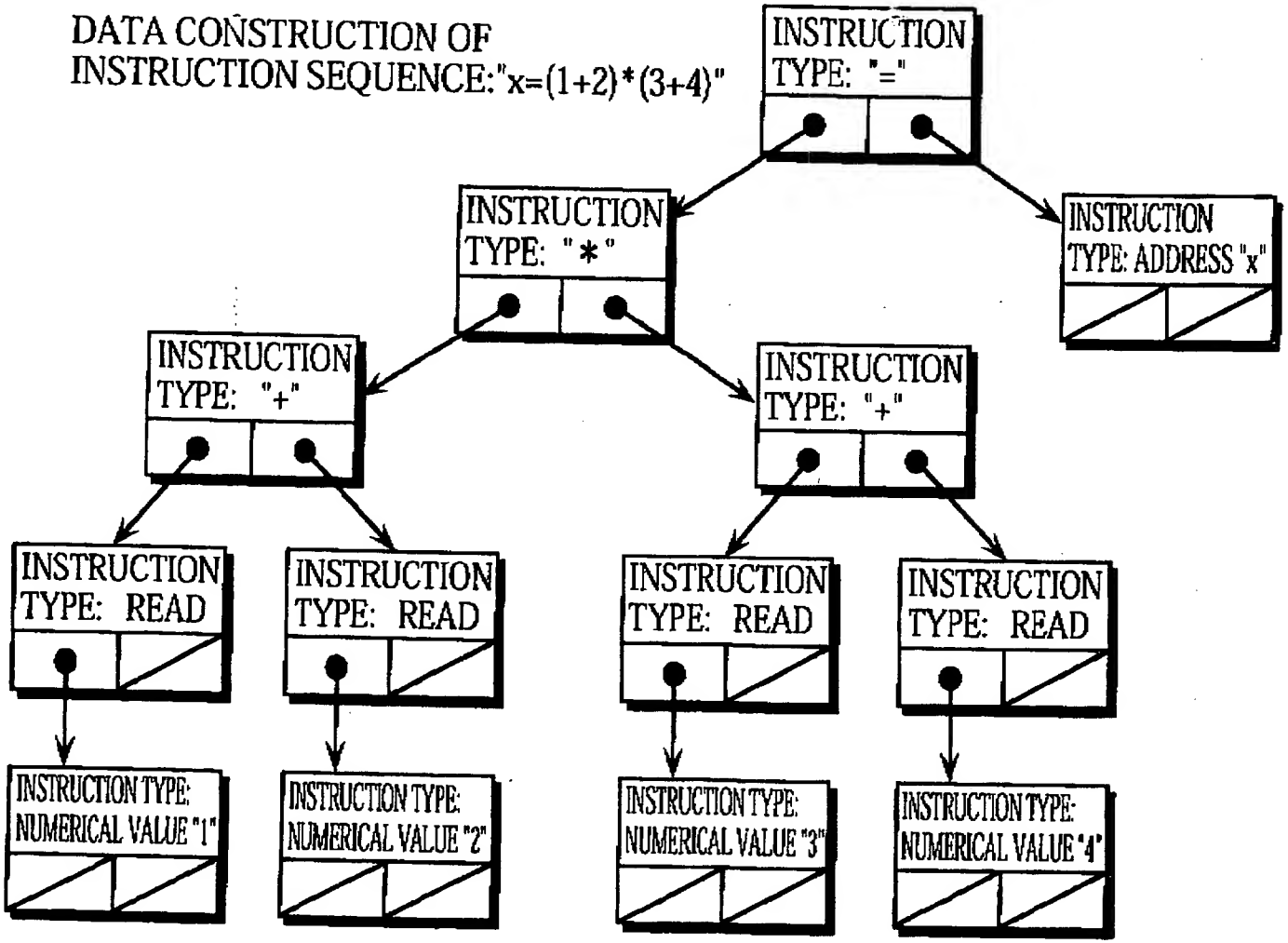


FIG. 54

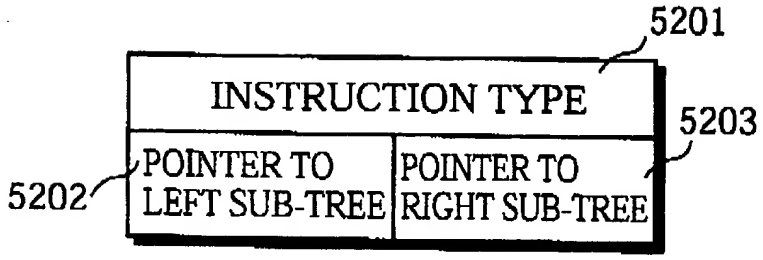

5201
INSTRUCTION TYPE
5203
5202
POINTER TO LEFT SUB-TREE
POINTER TO RIGHT SUB-TREE

FIG. 55

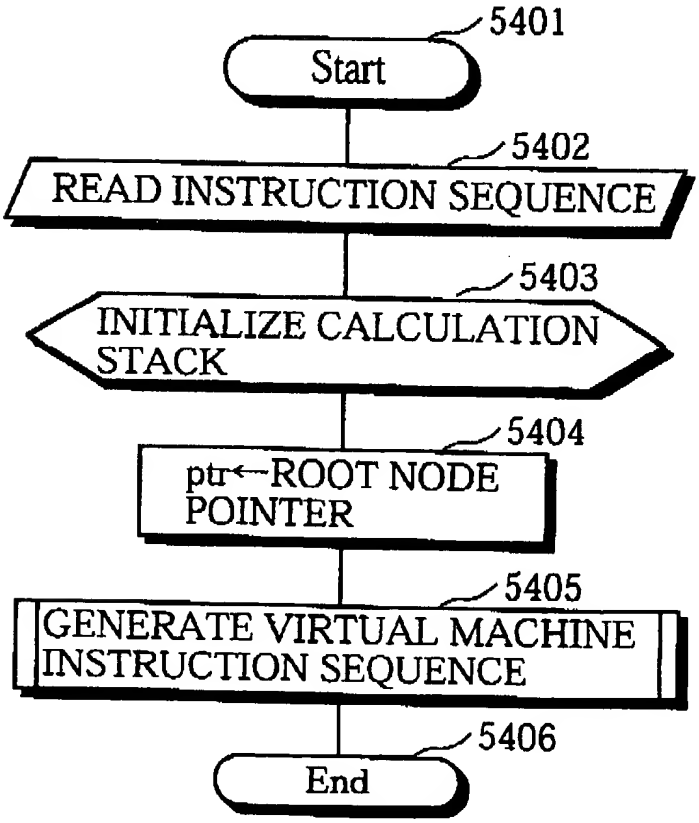

5401
Start
5402
READ INSTRUCTION SEQUENCE
5403
INITIALIZE CALCULATION STACK
5404
ptr←ROOT NODE POINTER
5405
GENERATE VIRTUAL MACHINE INSTRUCTION SEQUENCE
5406
End

FIG. 56

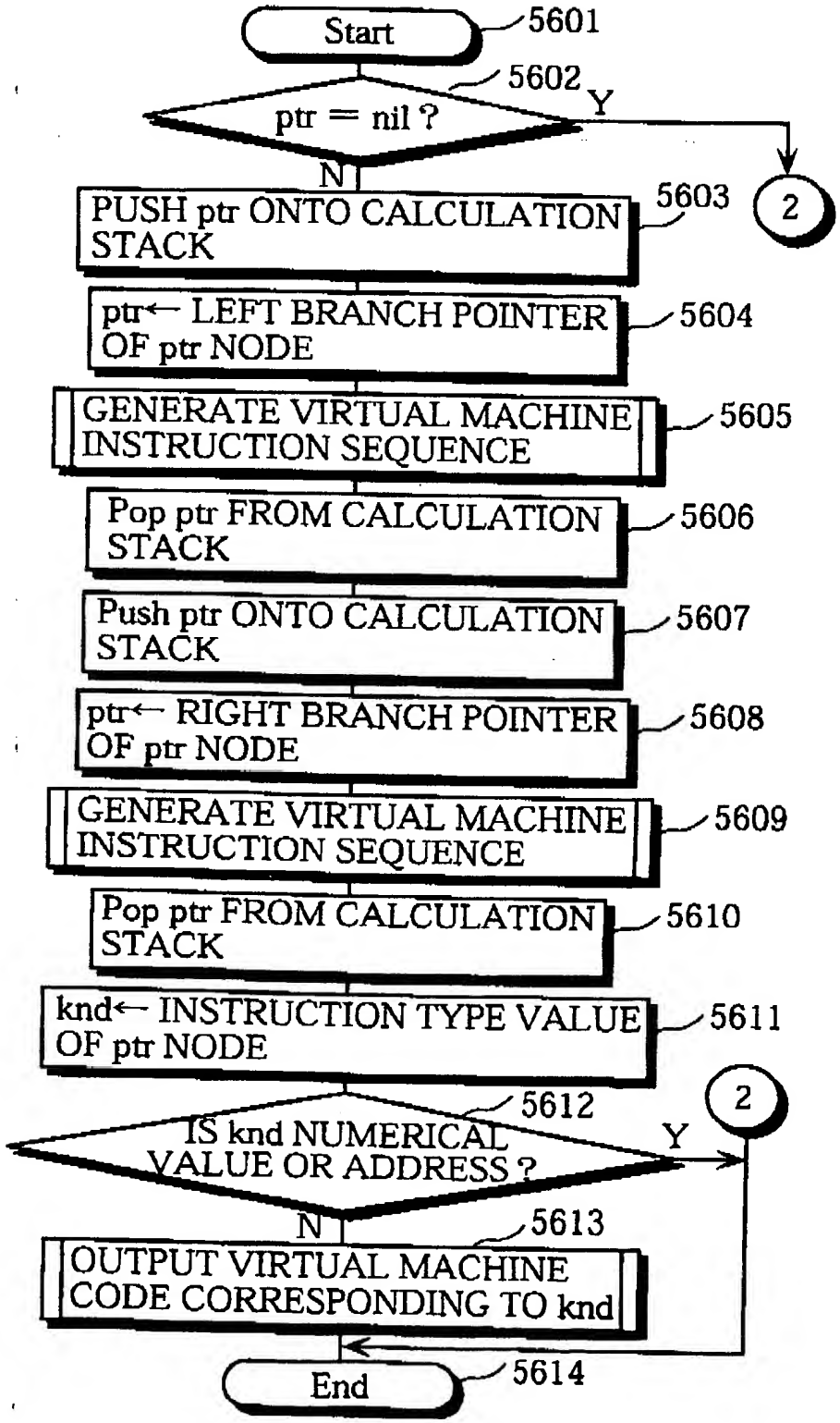

Start
5601
ptr = nil ?
5602
Y
N
2
PUSH ptr ONTO CALCULATION STACK
5603
ptr← LEFT BRANCH POINTER OF ptr NODE
5604
GENERATE VIRTUAL MACHINE INSTRUCTION SEQUENCE
5605
Pop ptr FROM CALCULATION STACK
5606
Push ptr ONTO CALCULATION STACK
5607
ptr← RIGHT BRANCH POINTER OF ptr NODE
5608
GENERATE VIRTUAL MACHINE INSTRUCTION SEQUENCE
5609
Pop ptr FROM CALCULATION STACK
5610
knd← INSTRUCTION TYPE VALUE OF ptr NODE
5611
IS knd NUMERICAL VALUE OR ADDRESS ?
5612
Y
2
N
OUTPUT VIRTUAL MACHINE CODE CORRESPONDING TO knd
5613
End
5614

FIG. 57A

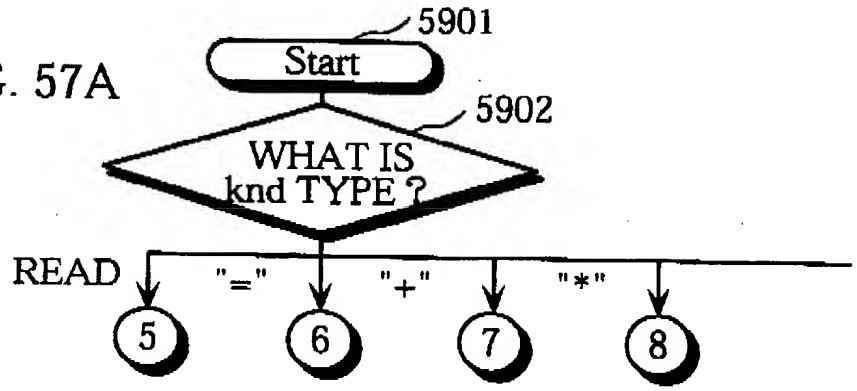


FIG. 57B

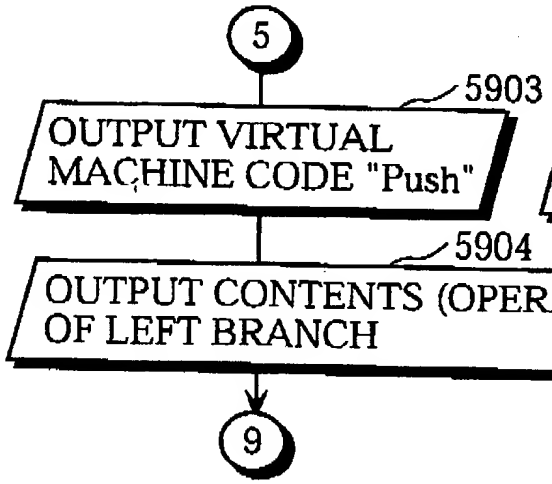


FIG. 57C

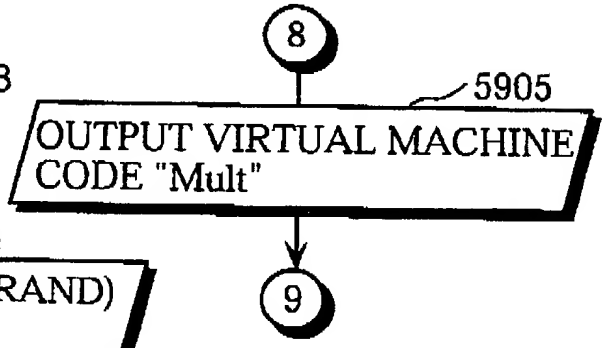


FIG. 57E

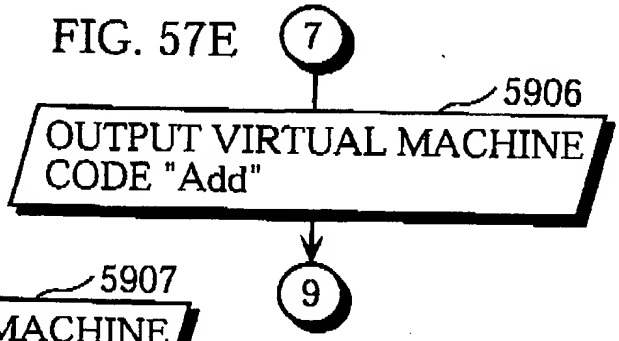
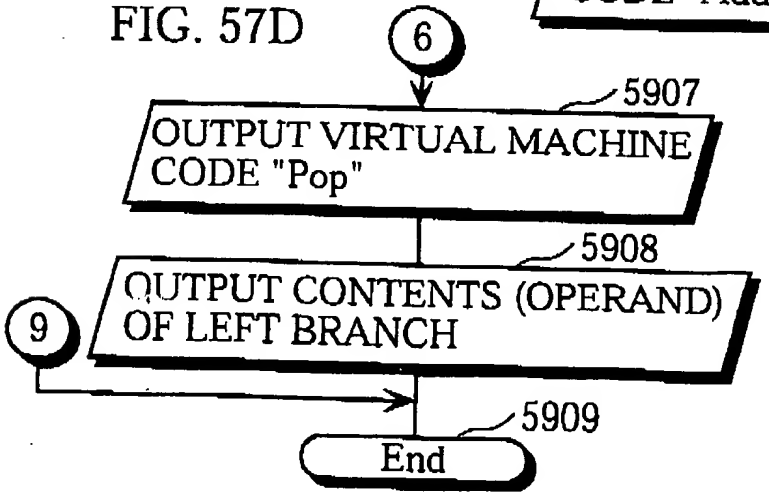


FIG. 57D



EP 2 282 260 A1

FIG. 58

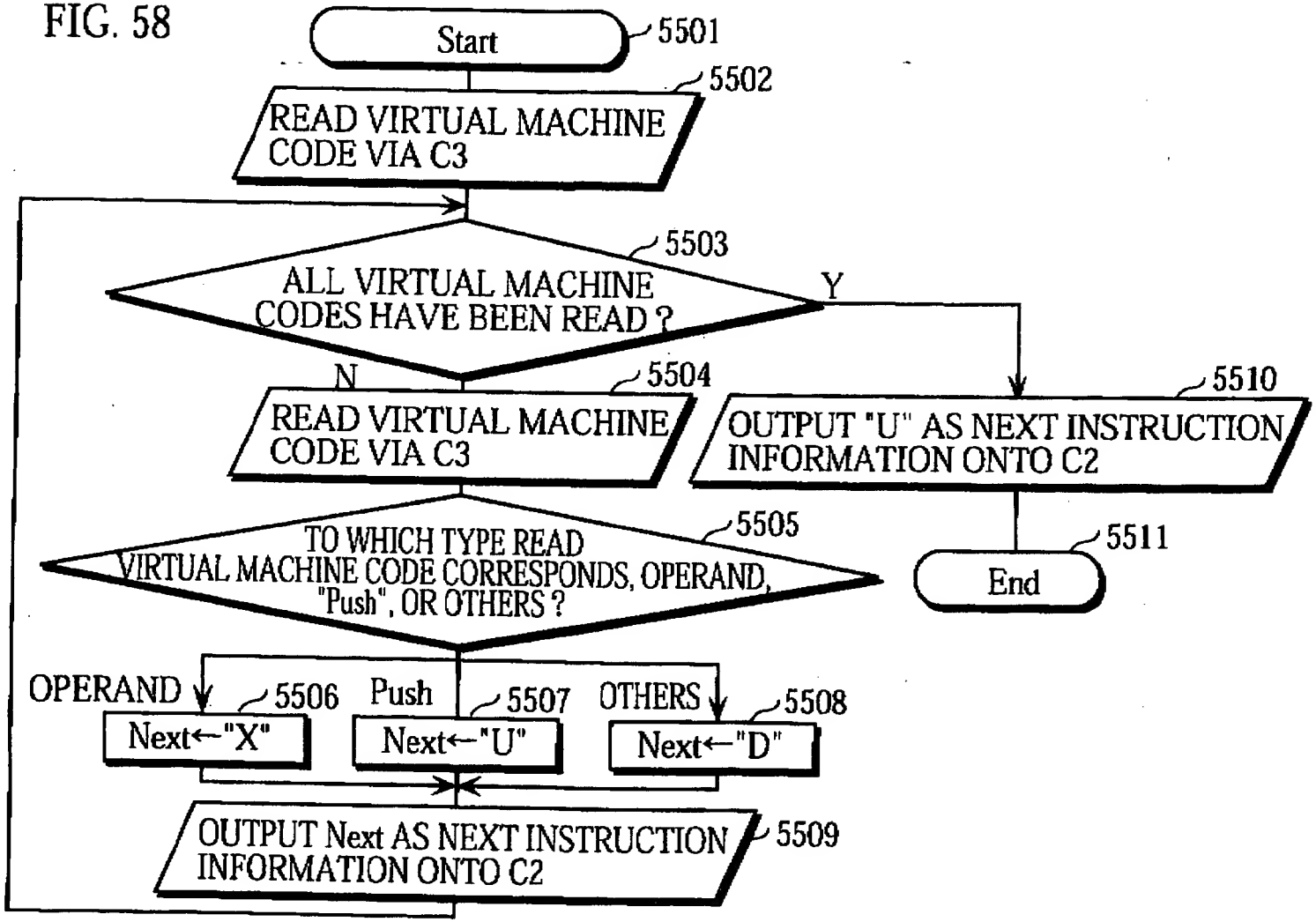


FIG. 59

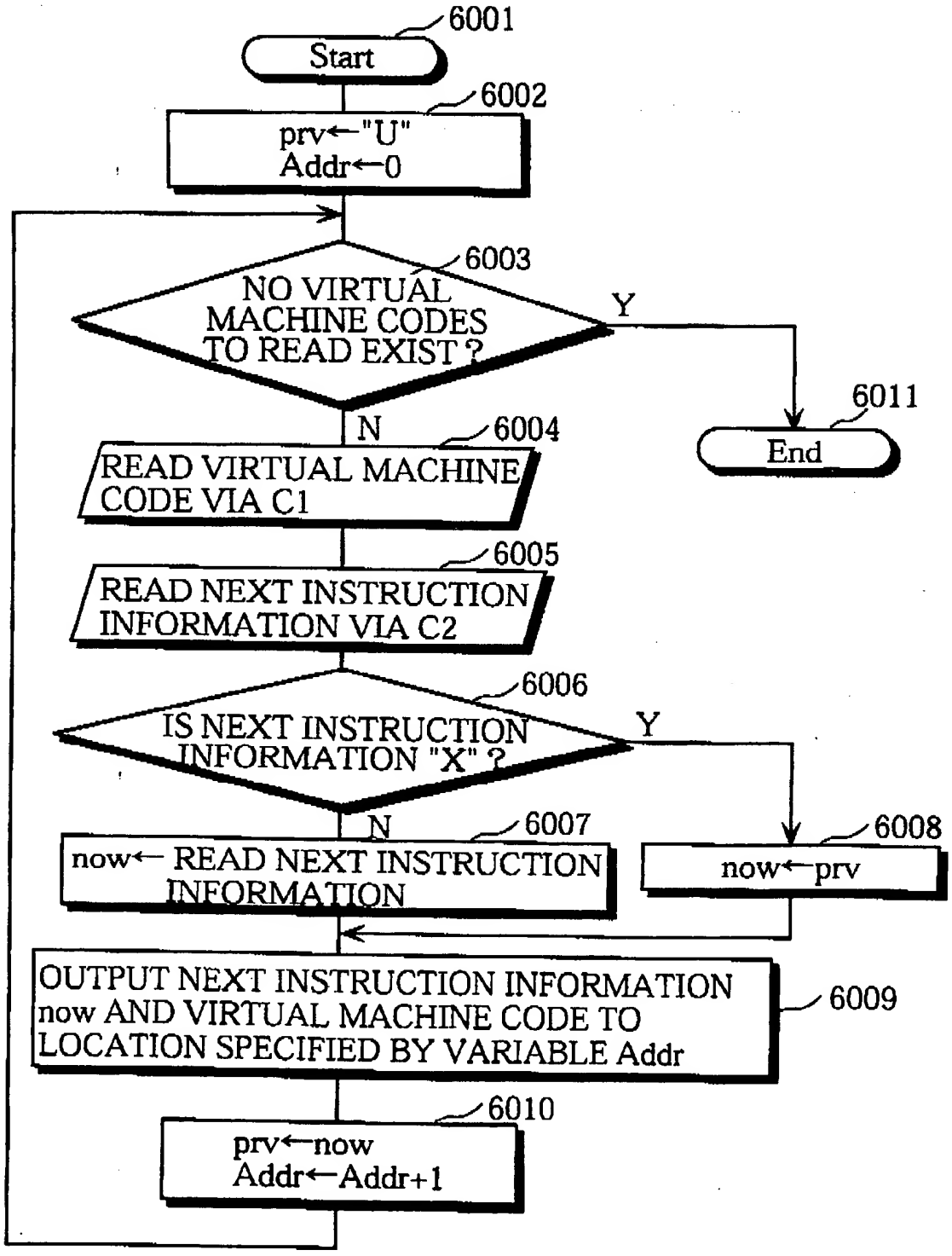

6001
Start
6002
prv←"U"
Addr←0
6003
NO VIRTUAL MACHINE CODES TO READ EXIST ?
Y
N
6011
End
6004
READ VIRTUAL MACHINE CODE VIA C1
6005
READ NEXT INSTRUCTION INFORMATION VIA C2
6006
IS NEXT INSTRUCTION INFORMATION "X" ?
Y
N
6007
now← READ NEXT INSTRUCTION INFORMATION
6008
now←prv
OUTPUT NEXT INSTRUCTION INFORMATION now AND VIRTUAL MACHINE CODE TO LOCATION SPECIFIED BY VARIABLE Addr
6009
6010
prv←now
Addr←Addr+1

FIG. 60

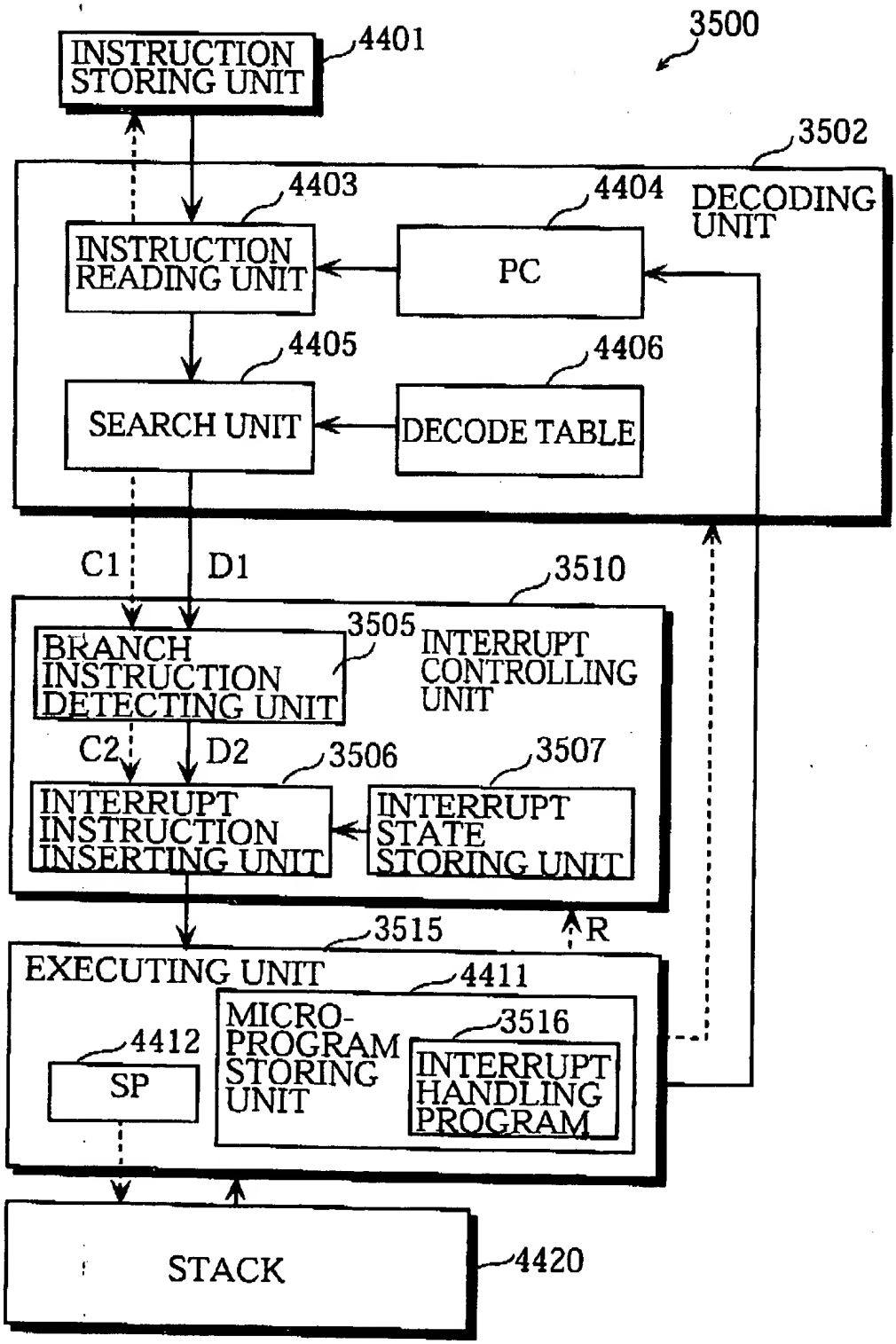

INSTRUCTION STORING UNIT
4401
3500
3502
DECODING UNIT
4403
INSTRUCTION READING UNIT
4404
PC
4405
SEARCH UNIT
4406
DECODE TABLE
C1
D1
3510
3505
BRANCH INSTRUCTION DETECTING UNIT
INTERRUPT CONTROLLING UNIT
C2
D2
3506
INTERRUPT INSTRUCTION INSERTING UNIT
3507
INTERRUPT STATE STORING UNIT
3515
R
EXECUTING UNIT
4411
MICRO-PROGRAM STORING UNIT
3516
INTERRUPT HANDLING PROGRAM
4412
SP
STACK
4420

FIG. 61

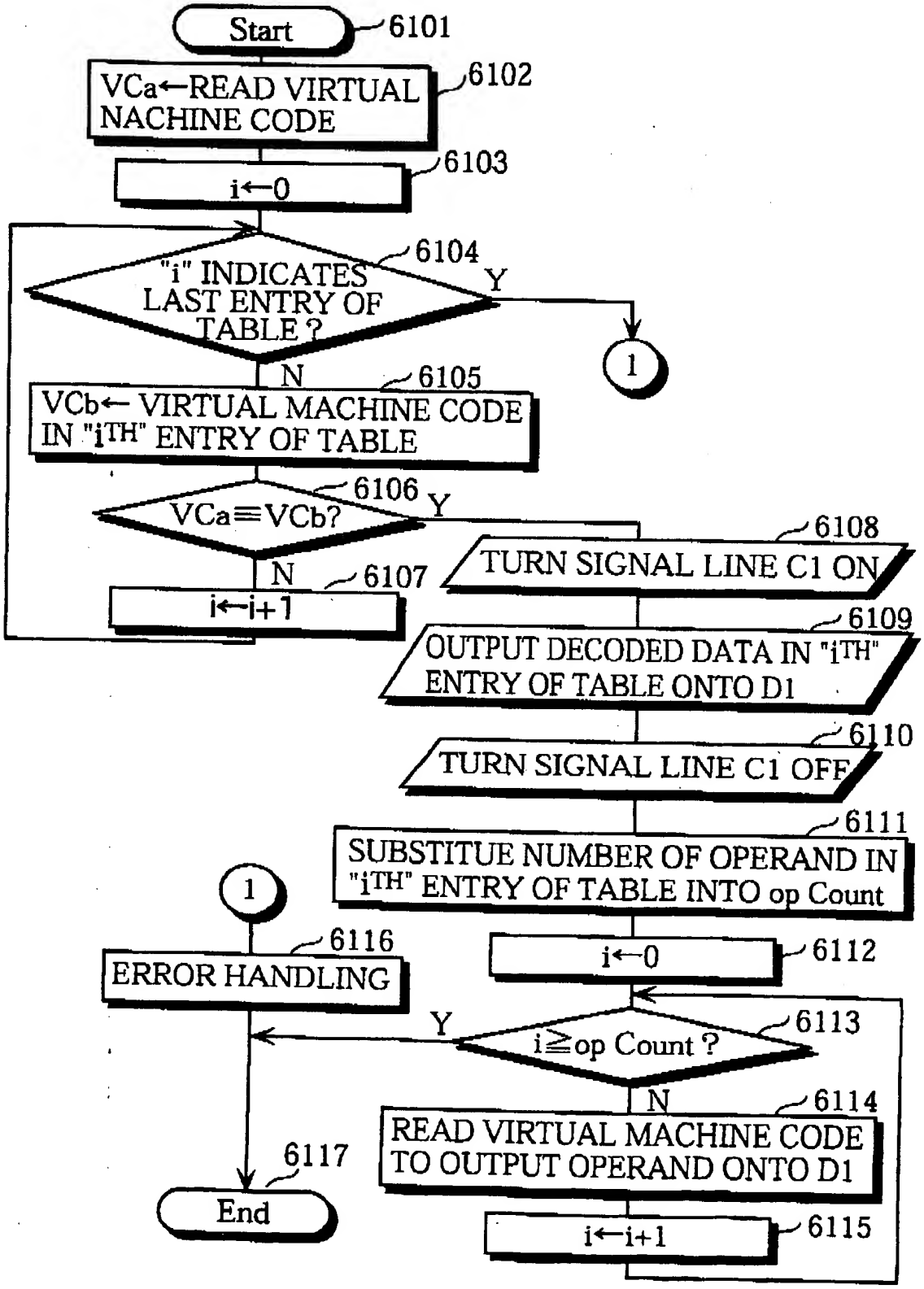

Start
6101
VCa←READ VIRTUAL NACHINE CODE
6102
i←0
6103
"i" INDICATES LAST ENTRY OF TABLE ?
6104
Y
N
1
VCb← VIRTUAL MACHINE CODE IN "iTH" ENTRY OF TABLE
6105
VCa≡VCb?
6106
Y
N
6107
i←i+1
6108
TURN SIGNAL LINE C1 ON
6109
OUTPUT DECODED DATA IN "iTH" ENTRY OF TABLE ONTO D1
6110
TURN SIGNAL LINE C1 OFF
6111
SUBSTITUE NUMBER OF OPERAND IN "iTH" ENTRY OF TABLE INTO op Count
1
6116
ERROR HANDLING
i←0
6112
Y
i≧op Count ?
6113
N
6114
READ VIRTUAL MACHINE CODE TO OUTPUT OPERAND ONTO D1
6117
End
i←i+1
6115

FIG. 62

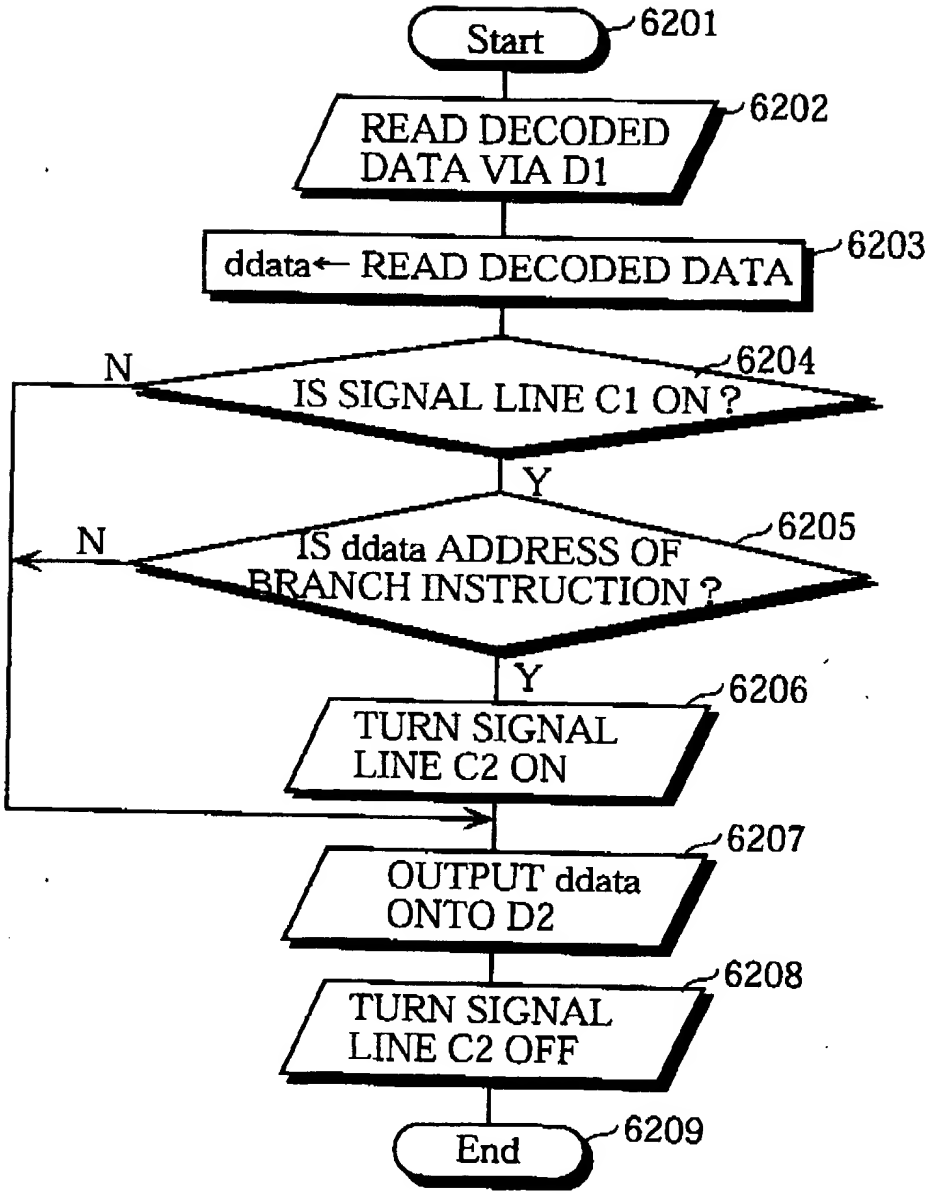

Start
6201
READ DECODED DATA VIA D1
6202
ddata← READ DECODED DATA
6203
N
IS SIGNAL LINE C1 ON ?
6204
Y
N
IS ddata ADDRESS OF BRANCH INSTRUCTION ?
6205
Y
TURN SIGNAL LINE C2 ON
6206
OUTPUT ddata ONTO D2
6207
TURN SIGNAL LINE C2 OFF
6208
End
6209

FIG. 63

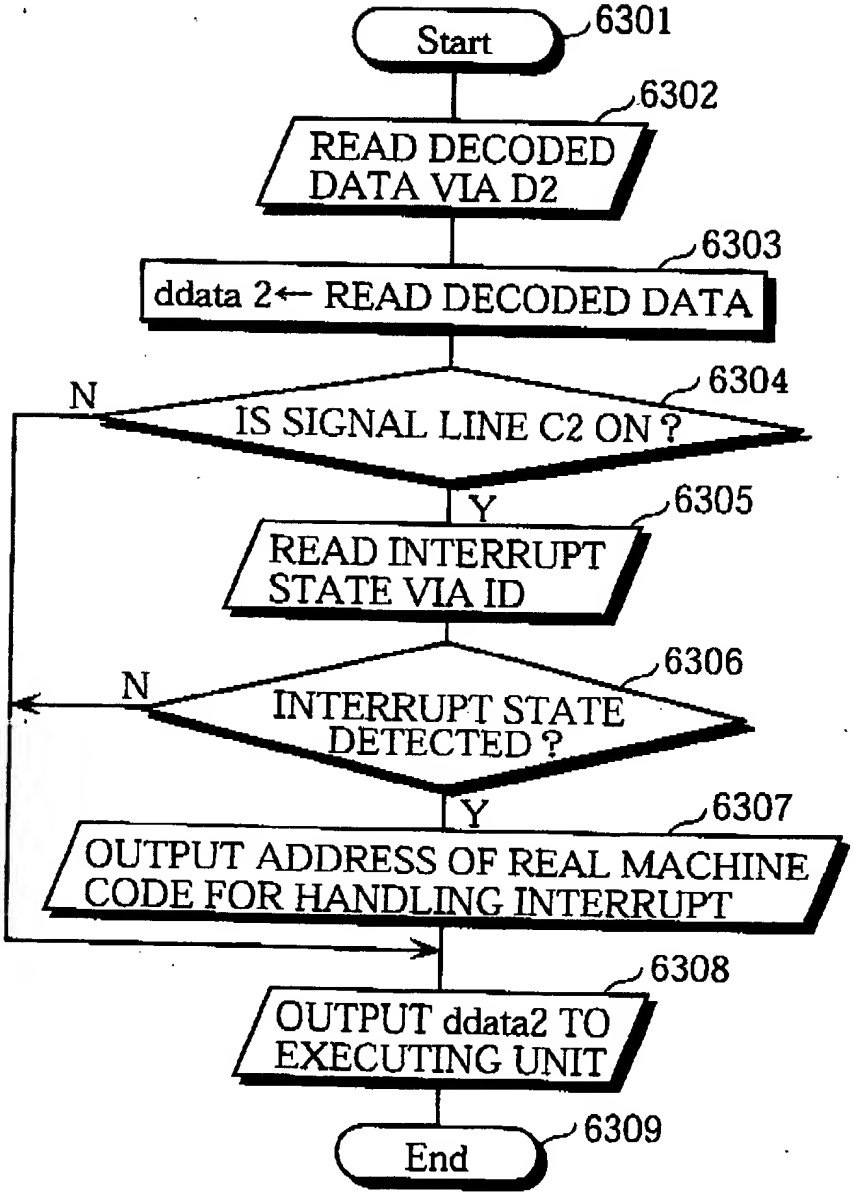

Start
6301
READ DECODED DATA VIA D2
6302
ddata 2← READ DECODED DATA
6303
IS SIGNAL LINE C2 ON ?
6304
N
Y
READ INTERRUPT STATE VIA ID
6305
INTERRUPT STATE DETECTED ?
6306
N
Y
OUTPUT ADDRESS OF REAL MACHINE CODE FOR HANDLING INTERRUPT
6307
OUTPUT ddata2 TO EXECUTING UNIT
6308
End
6309

FIG. 64

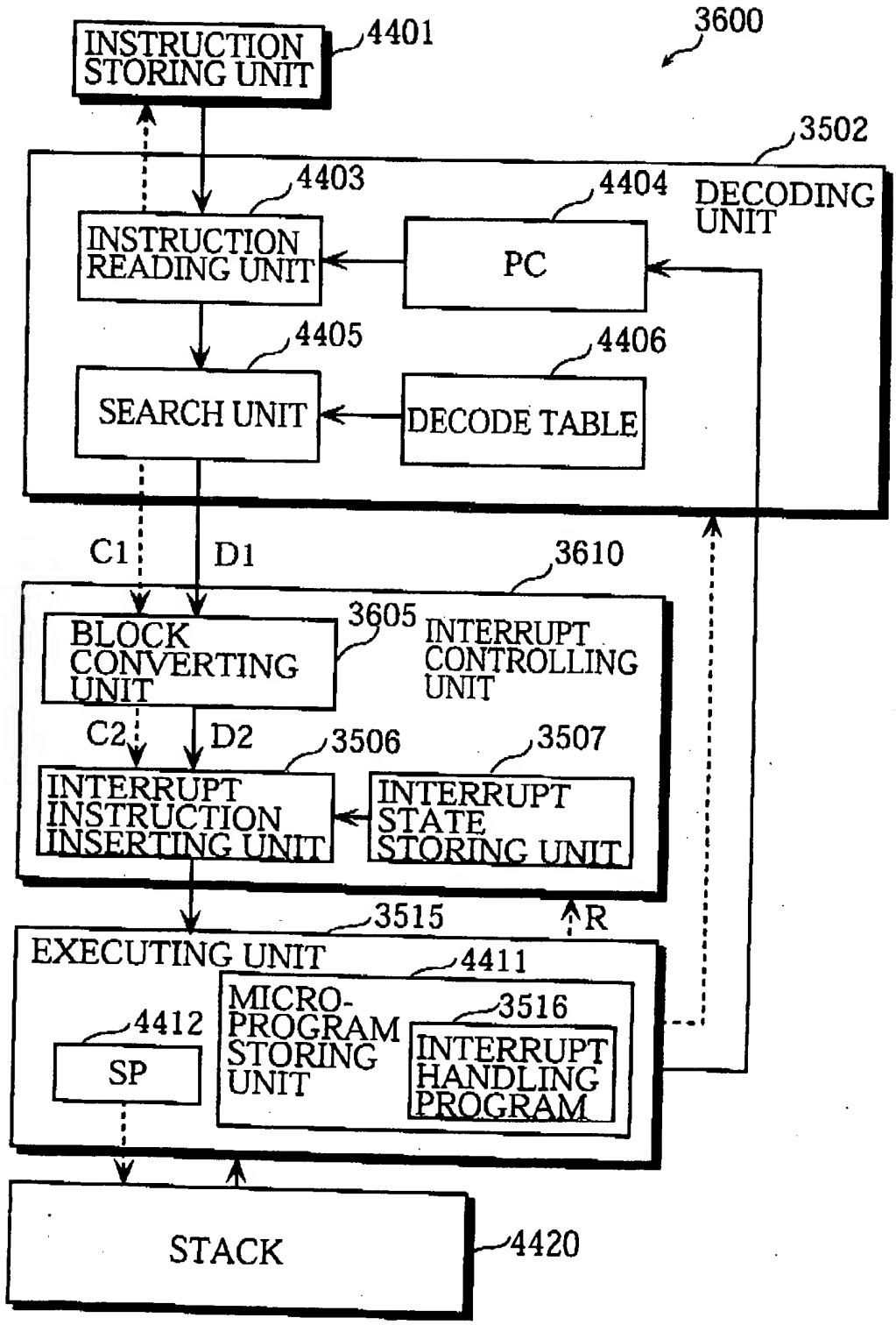

INSTRUCTION STORING UNIT
4401
3600
3502
4403
4404
DECODING UNIT
INSTRUCTION READING UNIT
PC
4405
4406
SEARCH UNIT
DECODE TABLE
C1
D1
3610
BLOCK CONVERTING UNIT
3605
INTERRUPT CONTROLLING UNIT
C2
D2
3506
3507
INTERRUPT INSTRUCTION INSERTING UNIT
INTERRUPT STATE STORING UNIT
3515
R
EXECUTING UNIT
4411
MICRO-PROGRAM STORING UNIT
3516
INTERRUPT HANDLING PROGRAM
4412
SP
STACK
4420

FIG. 65

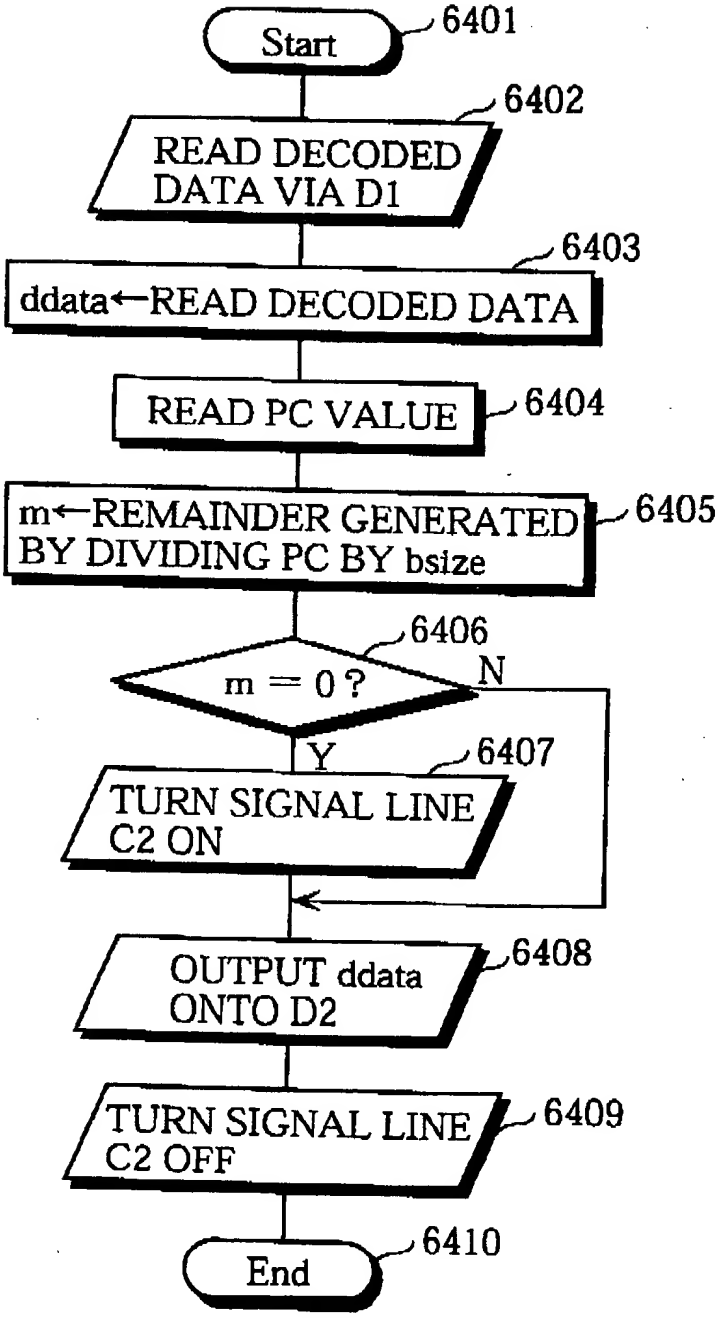

Start
6401
READ DECODED DATA VIA D1
6402
ddata←READ DECODED DATA
6403
READ PC VALUE
6404
m←REMAINDER GENERATED BY DIVIDING PC BY bsize
6405
m = 0 ?
6406
N
Y
TURN SIGNAL LINE C2 ON
6407
OUTPUT ddata ONTO D2
6408
TURN SIGNAL LINE C2 OFF
6409
End
6410

FIG. 66

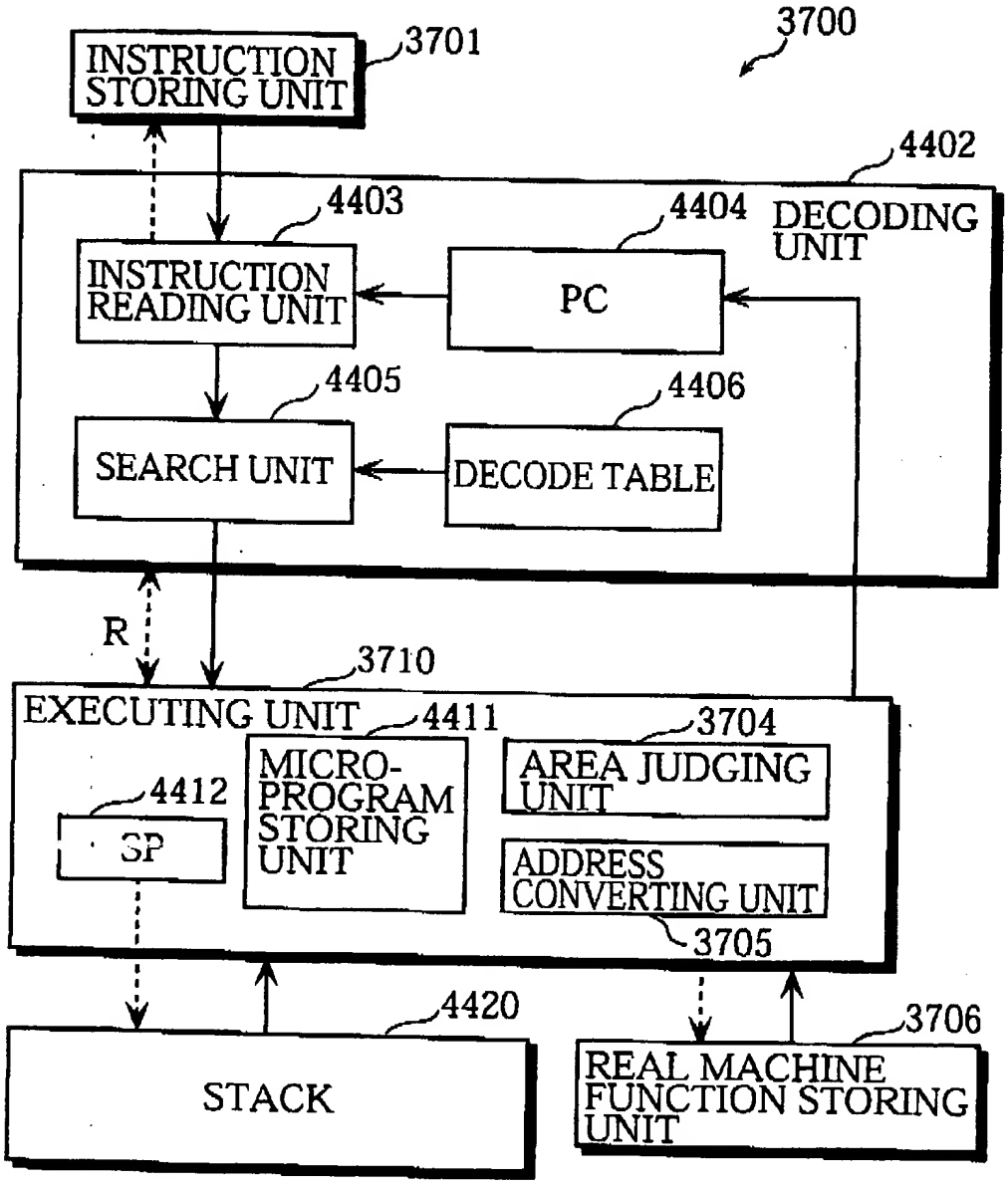

3700
INSTRUCTION STORING UNIT
3701
4402
DECODING UNIT
4403
INSTRUCTION READING UNIT
4404
PC
4405
SEARCH UNIT
4406
DECODE TABLE
R
3710
EXECUTING UNIT
4411
MICRO-PROGRAM STORING UNIT
3704
AREA JUDGING UNIT
4412
SP
ADDRESS CONVERTING UNIT
3705
4420
STACK
3706
REAL MACHINE FUNCTION STORING UNIT

FIG. 67

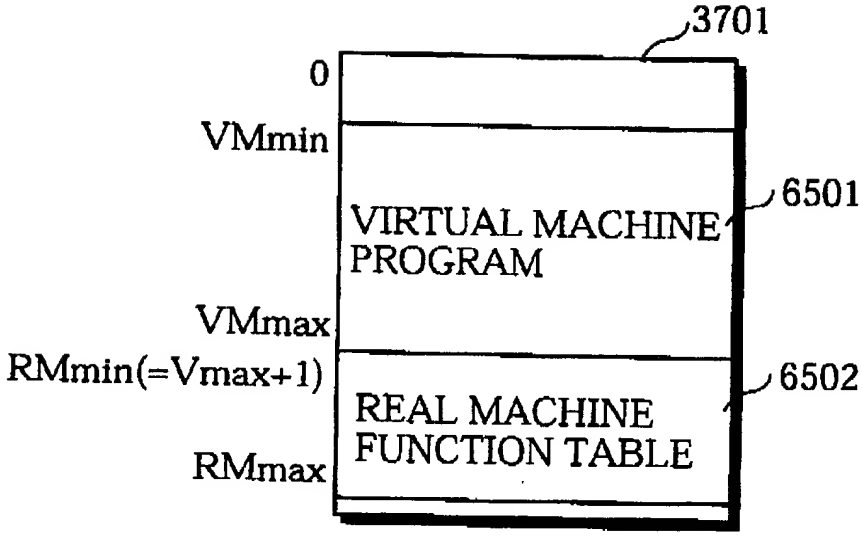

3701
0
VMmin
VIRTUAL MACHINE PROGRAM
6501
VMmax
RMmin(=Vmax+1)
REAL MACHINE FUNCTION TABLE
6502
RMmax

FIG. 68

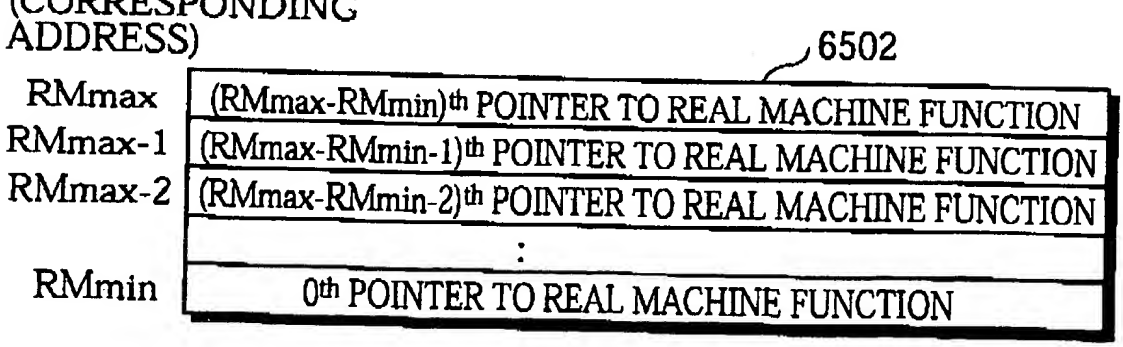
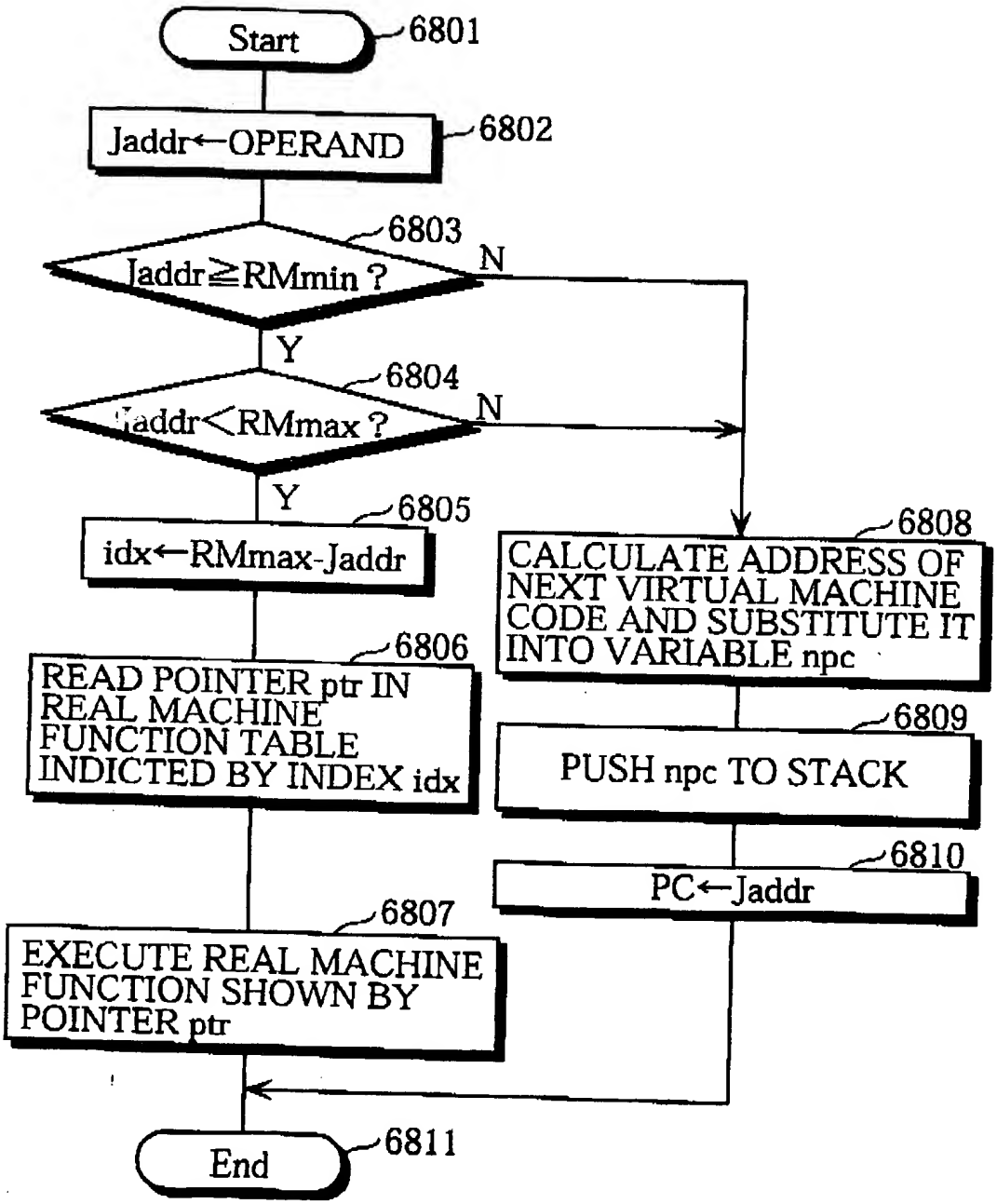

(CORRESPONDING ADDRESS)
6502
RMmax
(RMmax-RMmin)th POINTER TO REAL MACHINE FUNCTION
RMmax-1
(RMmax-RMmin-1)th POINTER TO REAL MACHINE FUNCTION
RMmax-2
(RMmax-RMmin-2)th POINTER TO REAL MACHINE FUNCTION
RMmin
0th POINTER TO REAL MACHINE FUNCTION

FIG. 69


Start
6801
Jaddr←OPERAND
6802
Jaddr≧RMmin ?
6803
N
Y
Jaddr<RMmax ?
6804
N
Y
idx←RMmax-Jaddr
6805
READ POINTER ptr IN REAL MACHINE FUNCTION TABLE INDICTED BY INDEX idx
6806
EXECUTE REAL MACHINE FUNCTION SHOWN BY POINTER ptr
6807
CALCULATE ADDRESS OF NEXT VIRTUAL MACHINE CODE AND SUBSTITUTE IT INTO VARIABLE npc
6808
PUSH npc TO STACK
6809
PC←Jaddr
6810
End
6811

EP 0 828 208 A2

FIG. 70

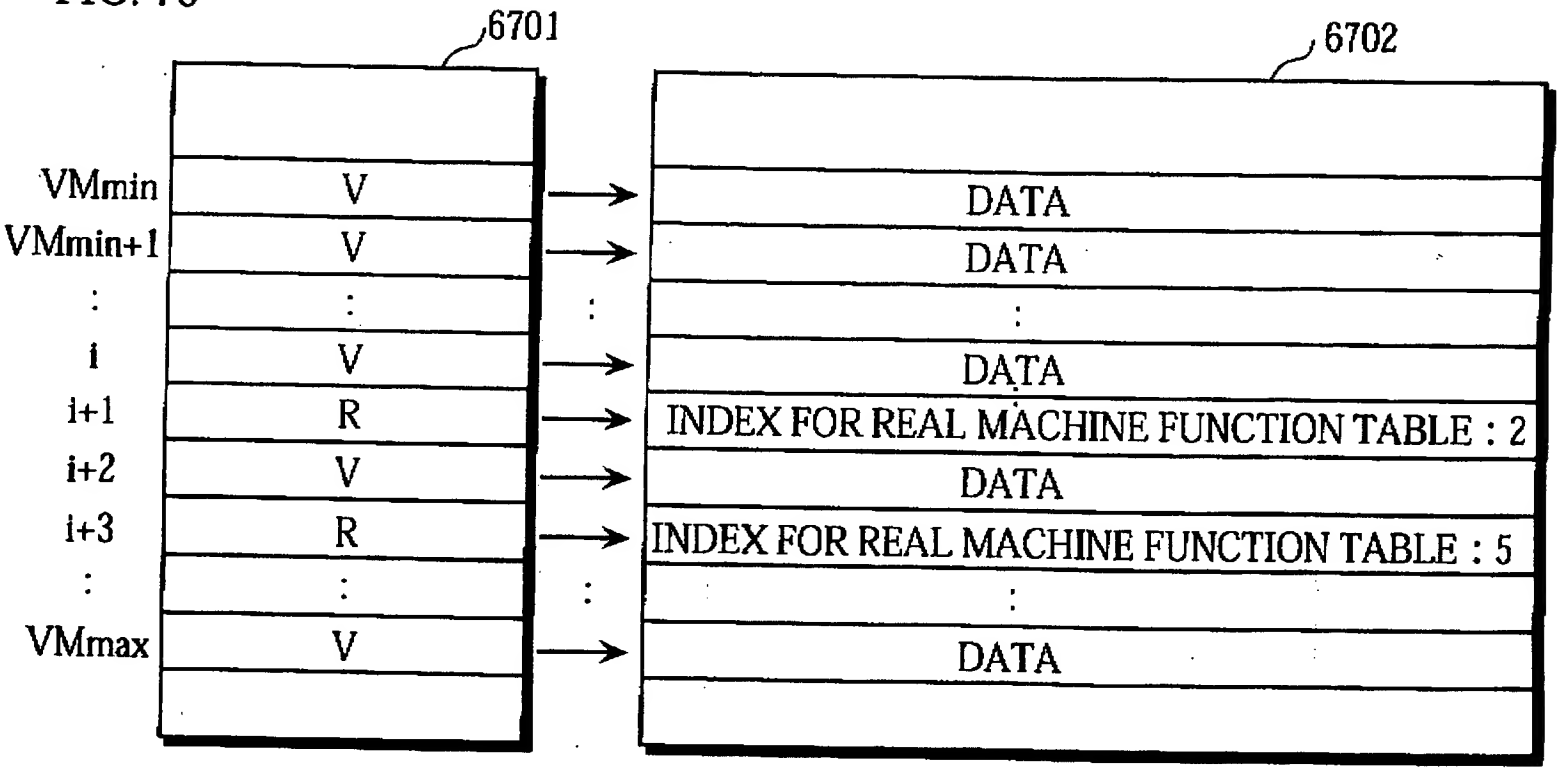


FIG. 71

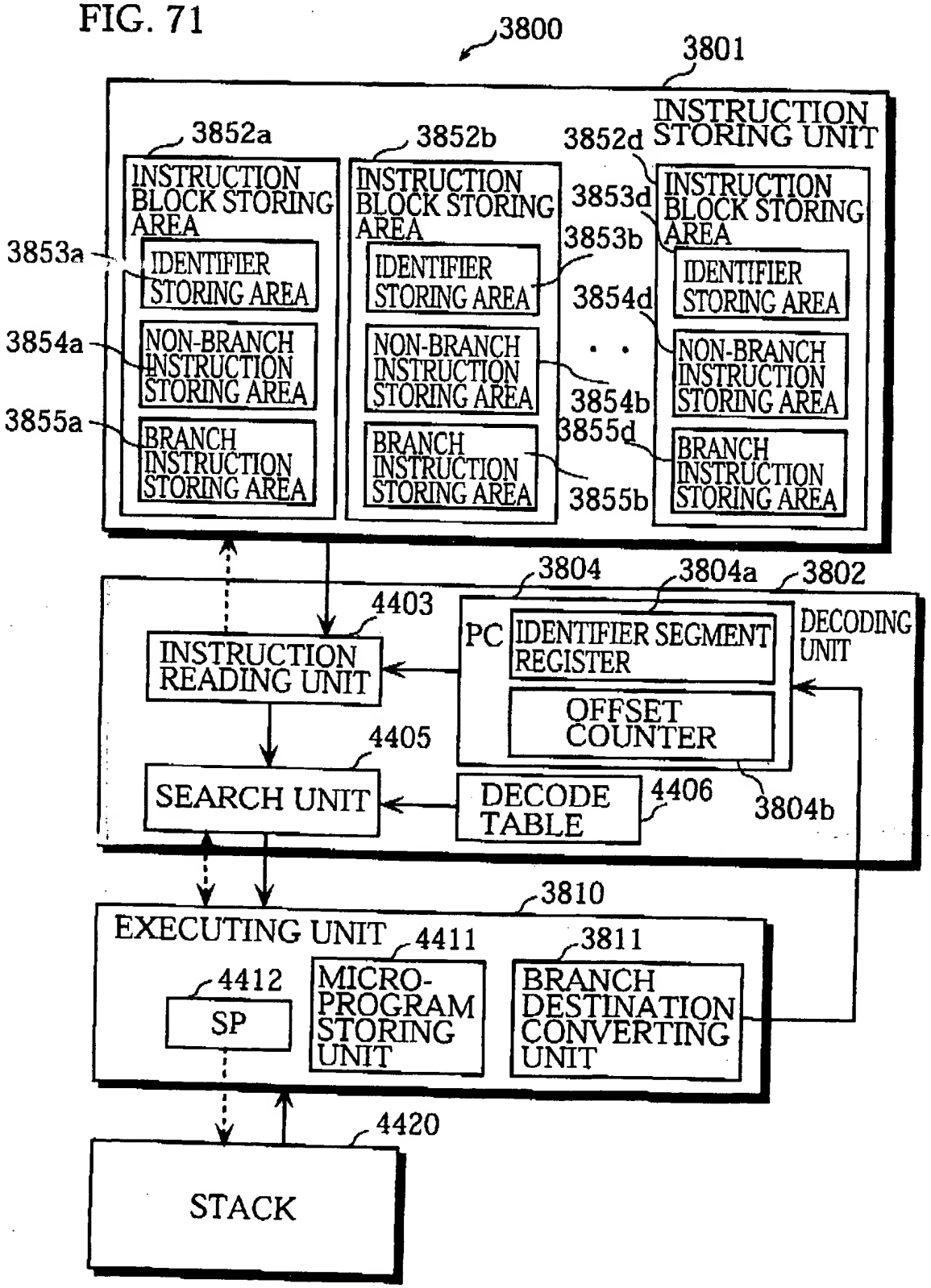

3800
3801
INSTRUCTION STORING UNIT
3852a
3852b
3852d
INSTRUCTION BLOCK STORING AREA
IDENTIFIER STORING AREA
NON-BRANCH INSTRUCTION STORING AREA
BRANCH INSTRUCTION STORING AREA
3853a
3854a
3855a
3853b
3854b
3855b
3853d
3854d
3855d
3804
3804a
3802
4403
PC
IDENTIFIER SEGMENT REGISTER
OFFSET COUNTER
DECODING UNIT
INSTRUCTION READING UNIT
4405
SEARCH UNIT
DECODE TABLE
4406
3804b
3810
EXECUTING UNIT
4411
3811
4412
SP
MICRO-PROGRAM STORING UNIT
BRANCH DESTINATION CONVERTING UNIT
4420
STACK

FIG. 72

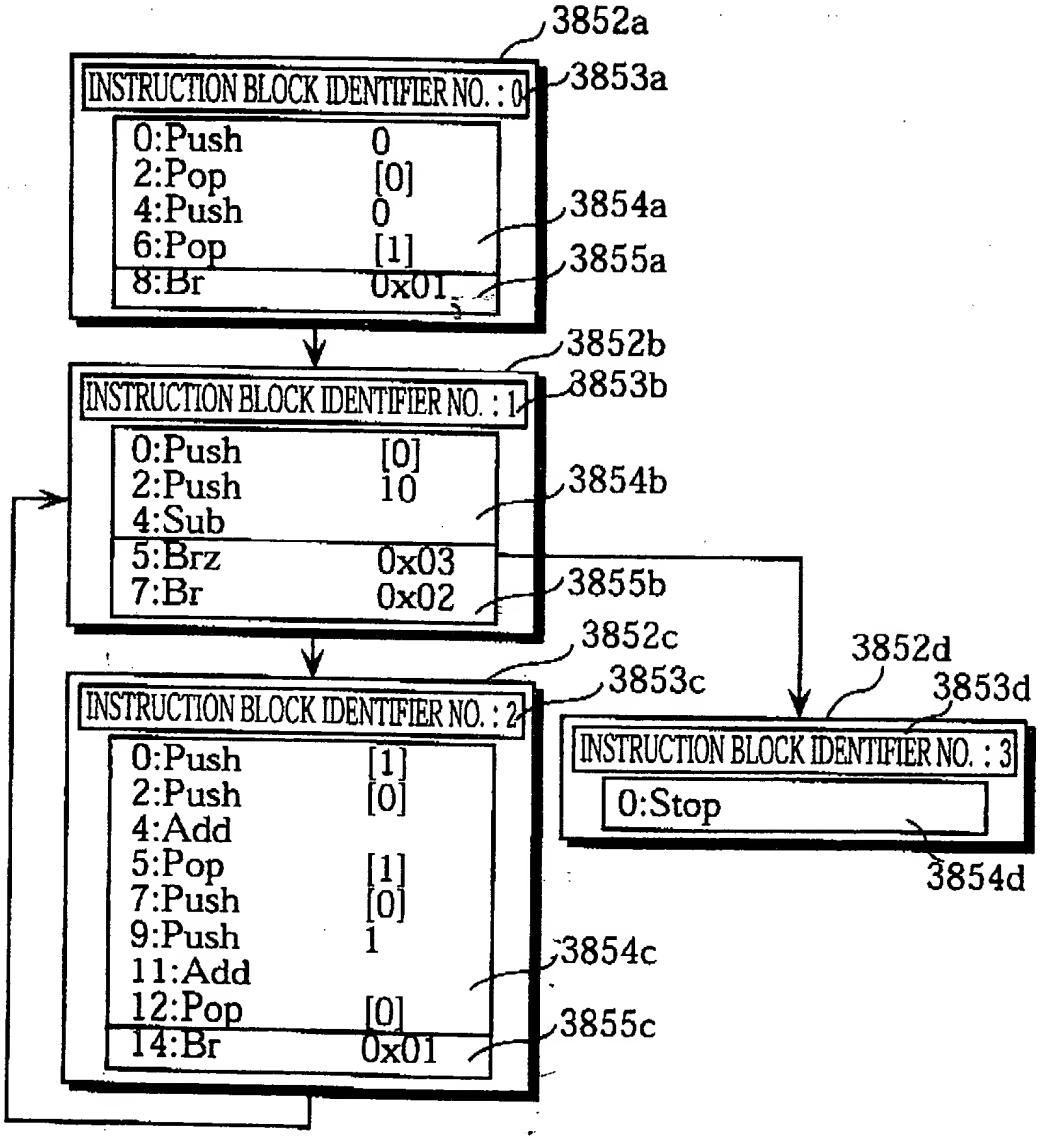

3852a
INSTRUCTION BLOCK IDENTIFIER NO. : 0
3853a
0:Push 0
2:Pop [0]
4:Push 0
6:Pop [1]
8:Br 0x01
3854a
3855a
3852b
INSTRUCTION BLOCK IDENTIFIER NO. : 1
3853b
0:Push [0]
2:Push 10
4:Sub
5:Brz 0x03
7:Br 0x02
3854b
3855b
3852c
INSTRUCTION BLOCK IDENTIFIER NO. : 2
3853c
0:Push [1]
2:Push [0]
4:Add
5:Pop [1]
7:Push [0]
9:Push 1
11:Add
12:Pop [0]
14:Br 0x01
3854c
3855c
3852d
3853d
INSTRUCTION BLOCK IDENTIFIER NO. : 3
0:Stop
3854d

FIG. 73

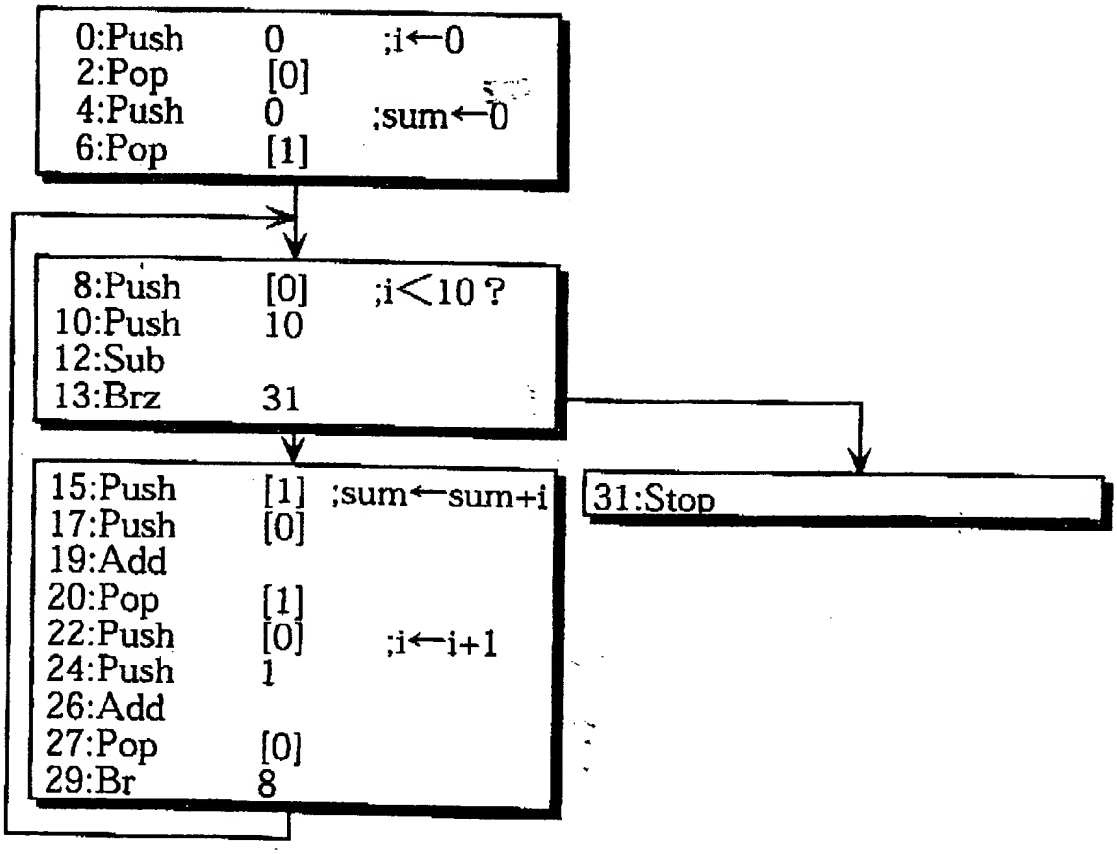

0:Push 0 ;i←0
2:Pop [0]
4:Push 0 ;sum←0
6:Pop [1]
8:Push [0] ;i<10 ?
10:Push 10
12:Sub
13:Brz 31
15:Push [1] ;sum←sum+i
17:Push [0]
19:Add
20:Pop [1]
22:Push [0] ;i←i+1
24:Push 1
26:Add
27:Pop [0]
29:Br 8
31:Stop

FIG. 74

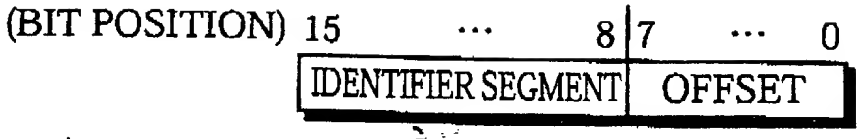

(BIT POSITION) 15 ··· 8 7 ··· 0
IDENTIFIER SEGMENT
OFFSET

FIG. 75

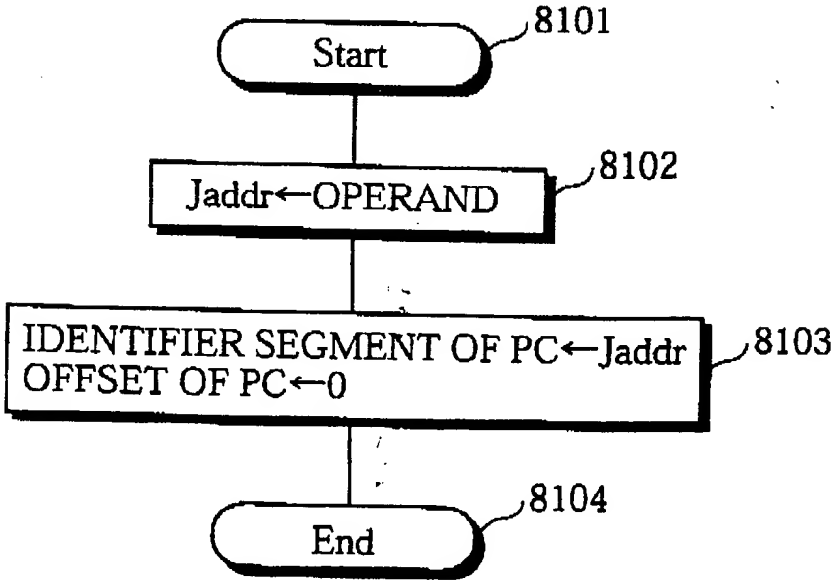

Start
8101
Jaddr←OPERAND
8102
IDENTIFIER SEGMENT OF PC←Jaddr
OFFSET OF PC←0
8103
End
8104

FIG. 7B

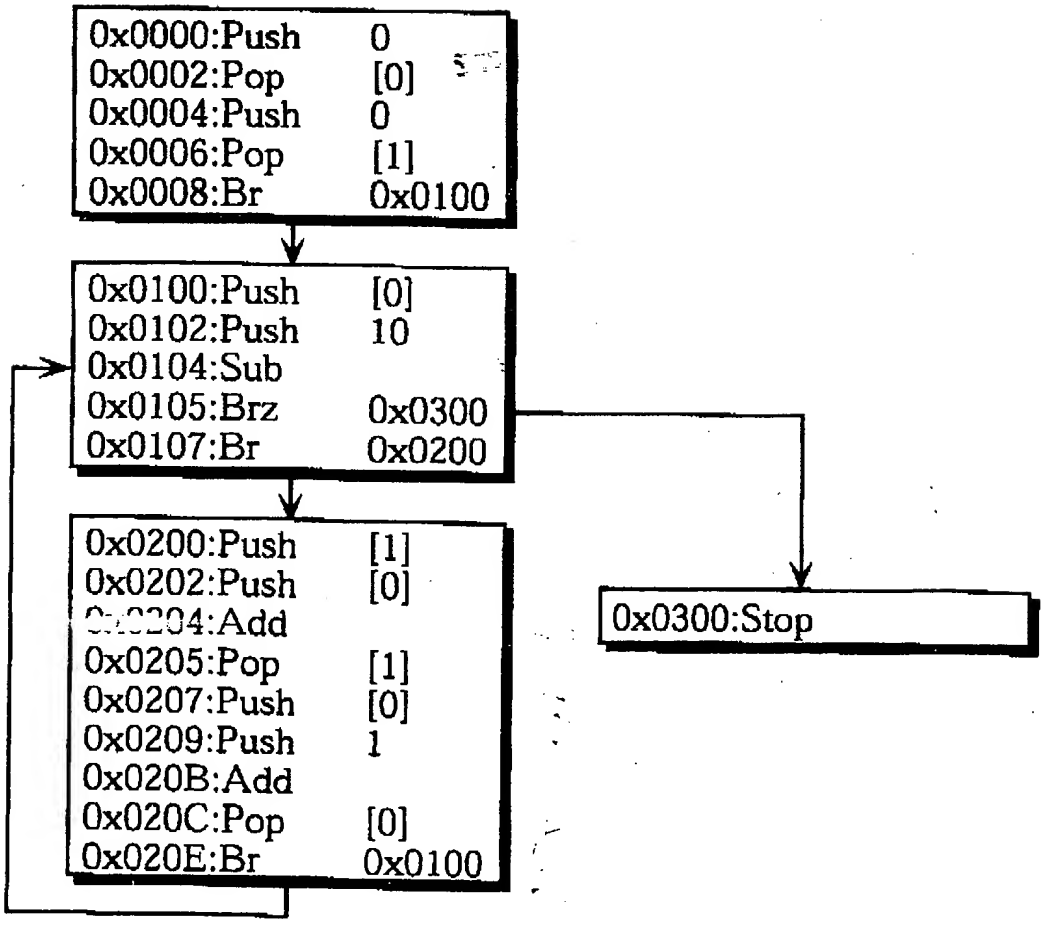
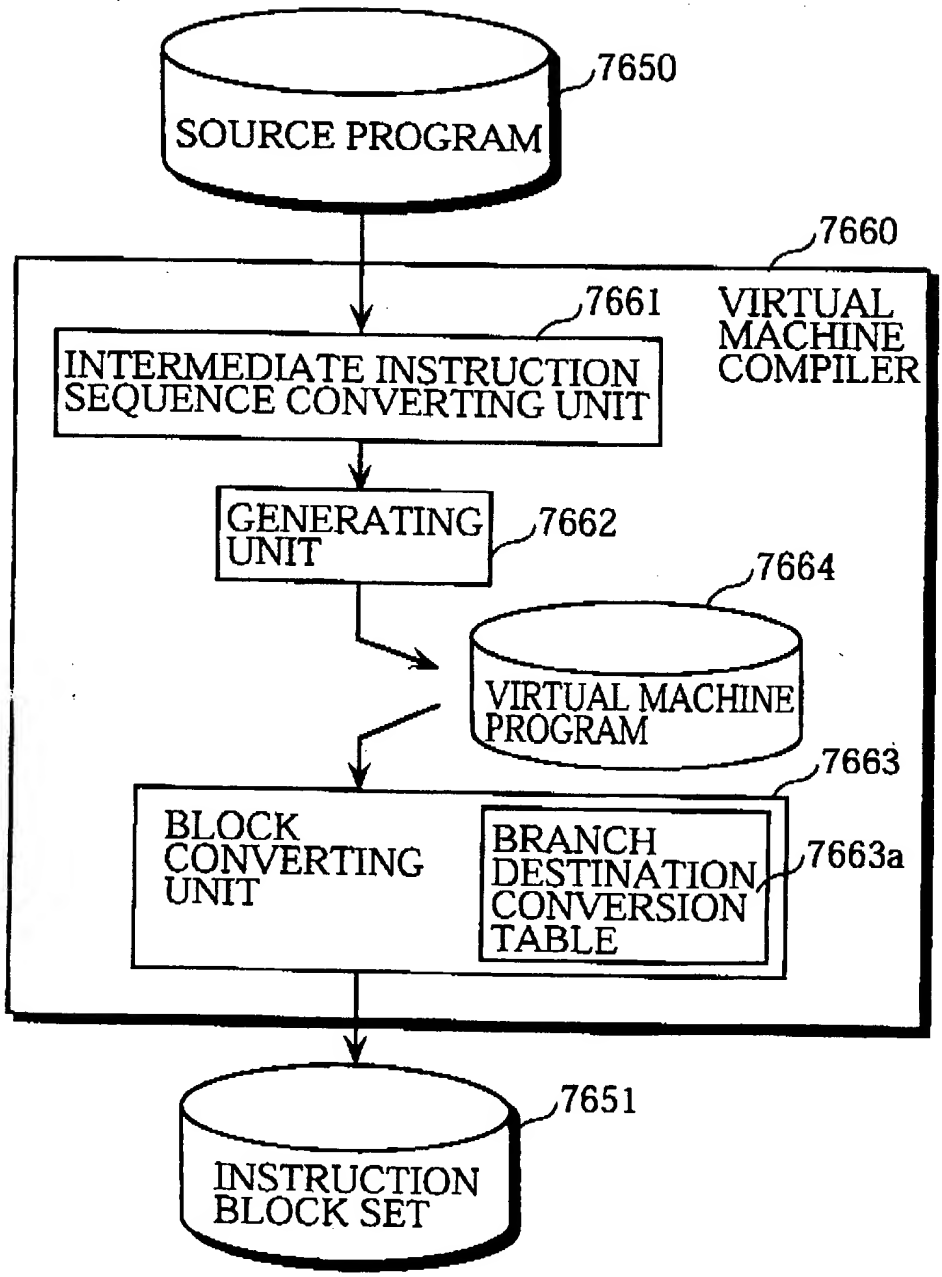

0x0000:Push 0
0x0002:Pop [0]
0x0004:Push 0
0x0006:Pop [1]
0x0008:Br 0x0100
0x0100:Push [0]
0x0102:Push 10
0x0104:Sub
0x0105:Brz 0x0300
0x0107:Br 0x0200
0x0200:Push [1]
0x0202:Push [0]
0x0204:Add
0x0205:Pop [1]
0x0207:Push [0]
0x0209:Push 1
0x020B:Add
0x020C:Pop [0]
0x020E:Br 0x0100
0x0300:Stop

FIG. 77


SOURCE PROGRAM
7650
7660
VIRTUAL MACHINE COMPILER
7661
INTERMEDIATE INSTRUCTION SEQUENCE CONVERTING UNIT
GENERATING UNIT
7662
7664
VIRTUAL MACHINE PROGRAM
7663
BLOCK CONVERTING UNIT
BRANCH DESTINATION CONVERSION TABLE
7663a
INSTRUCTION BLOCK SET
7651

6088040 E928B260

FIG. 78

7663a

0	CODE POSITION :	REGISTRATION FLAG :	REFERENCE POSITION OFFSET :	REFERENCE POSITION IDENTIFIER :
1	CODE POSITION :	REGISTRATION FLAG :	REFERENCE POSITION OFFSET :	REFERENCE POSITION IDENTIFIER :
2	CODE POSITION :	REGISTRATION FLAG :	REFERENCE POSITION OFFSET :	REFERENCE POSITION IDENTIFIER :
3	CODE POSITION :	REGISTRATION FLAG :	REFERENCE POSITION OFFSET :	REFERENCE POSITION IDENTIFIER :
4	CODE POSITION :	REGISTRATION FLAG :	REFERENCE POSITION OFFSET :	REFERENCE POSITION IDENTIFIER :
⋮				
Rcount-1	CODE POSITION :	REGISTRATION FLAG :	REFERENCE POSITION OFFSET :	REFERENCE POSITION IDENTIFIER :

FIG. 79

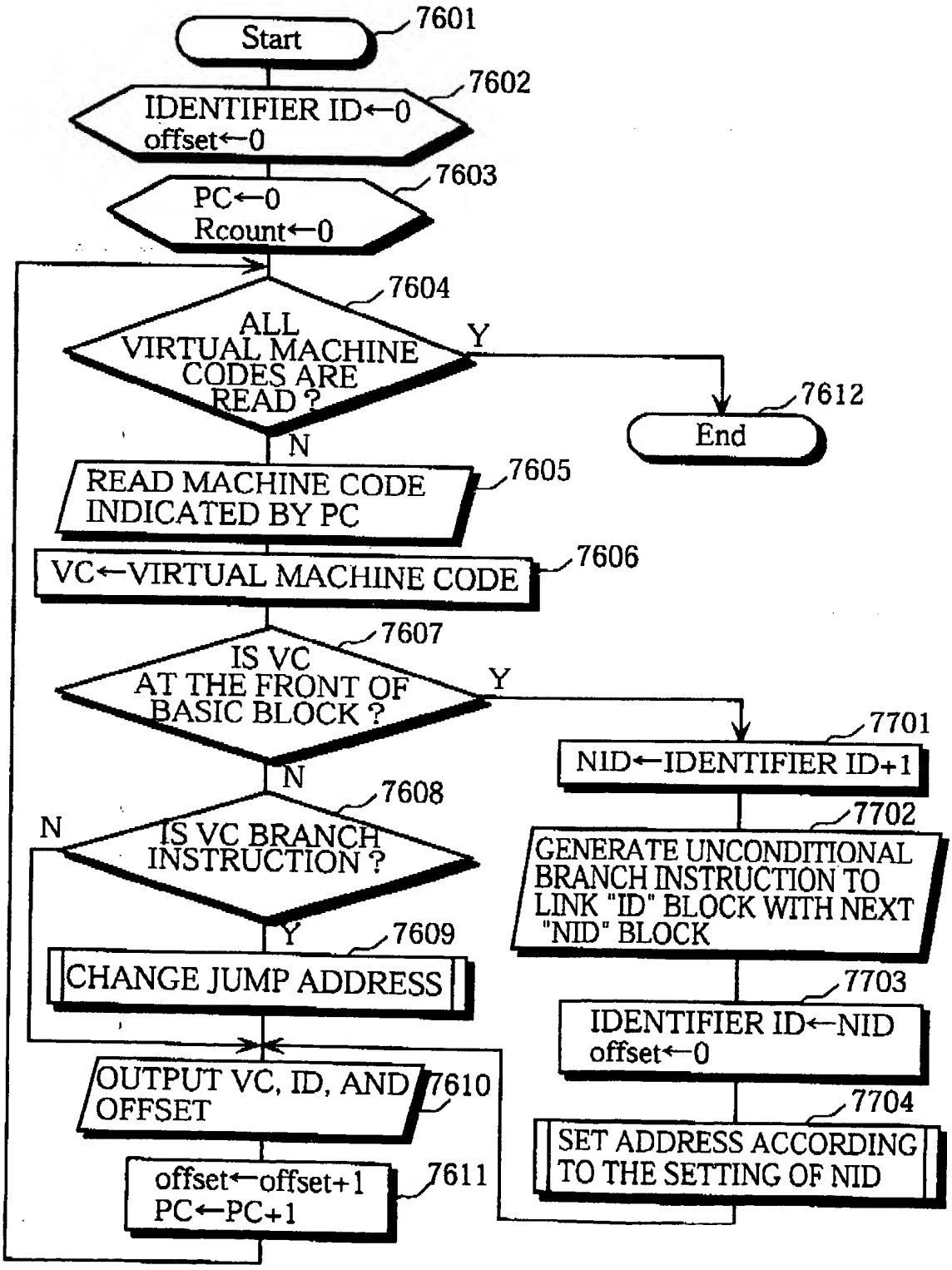

Start
7601
IDENTIFIER ID←0
offset←0
7602
PC←0
Rcount←0
7603
ALL VIRTUAL MACHINE CODES ARE READ ?
7604
Y
End
7612
N
READ MACHINE CODE INDICATED BY PC
7605
VC←VIRTUAL MACHINE CODE
7606
IS VC AT THE FRONT OF BASIC BLOCK ?
7607
Y
NID←IDENTIFIER ID+1
7701
N
IS VC BRANCH INSTRUCTION ?
7608
N
GENERATE UNCONDITIONAL BRANCH INSTRUCTION TO LINK "ID" BLOCK WITH NEXT "NID" BLOCK
7702
Y
CHANGE JUMP ADDRESS
7609
IDENTIFIER ID←NID
offset←0
7703
OUTPUT VC, ID, AND OFFSET
7610
SET ADDRESS ACCORDING TO THE SETTING OF NID
7704
offset←offset+1
PC←PC+1
7611

FIG. 80

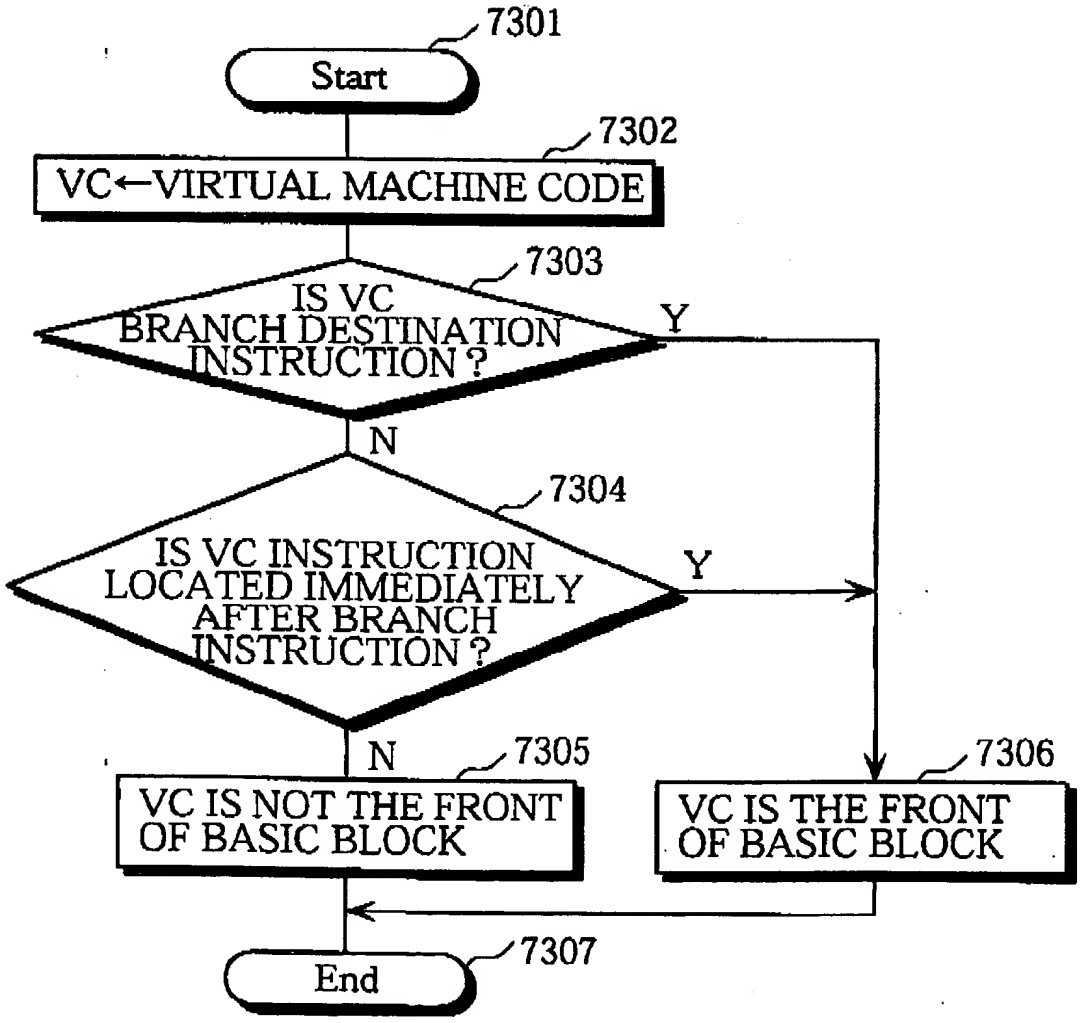

7301
Start
7302
VC←VIRTUAL MACHINE CODE
7303
IS VC BRANCH DESTINATION INSTRUCTION ?
Y
N
7304
IS VC INSTRUCTION LOCATED IMMEDIATELY AFTER BRANCH INSTRUCTION ?
Y
N
7305
VC IS NOT THE FRONT OF BASIC BLOCK
7306
VC IS THE FRONT OF BASIC BLOCK
7307
End

FIG. 81

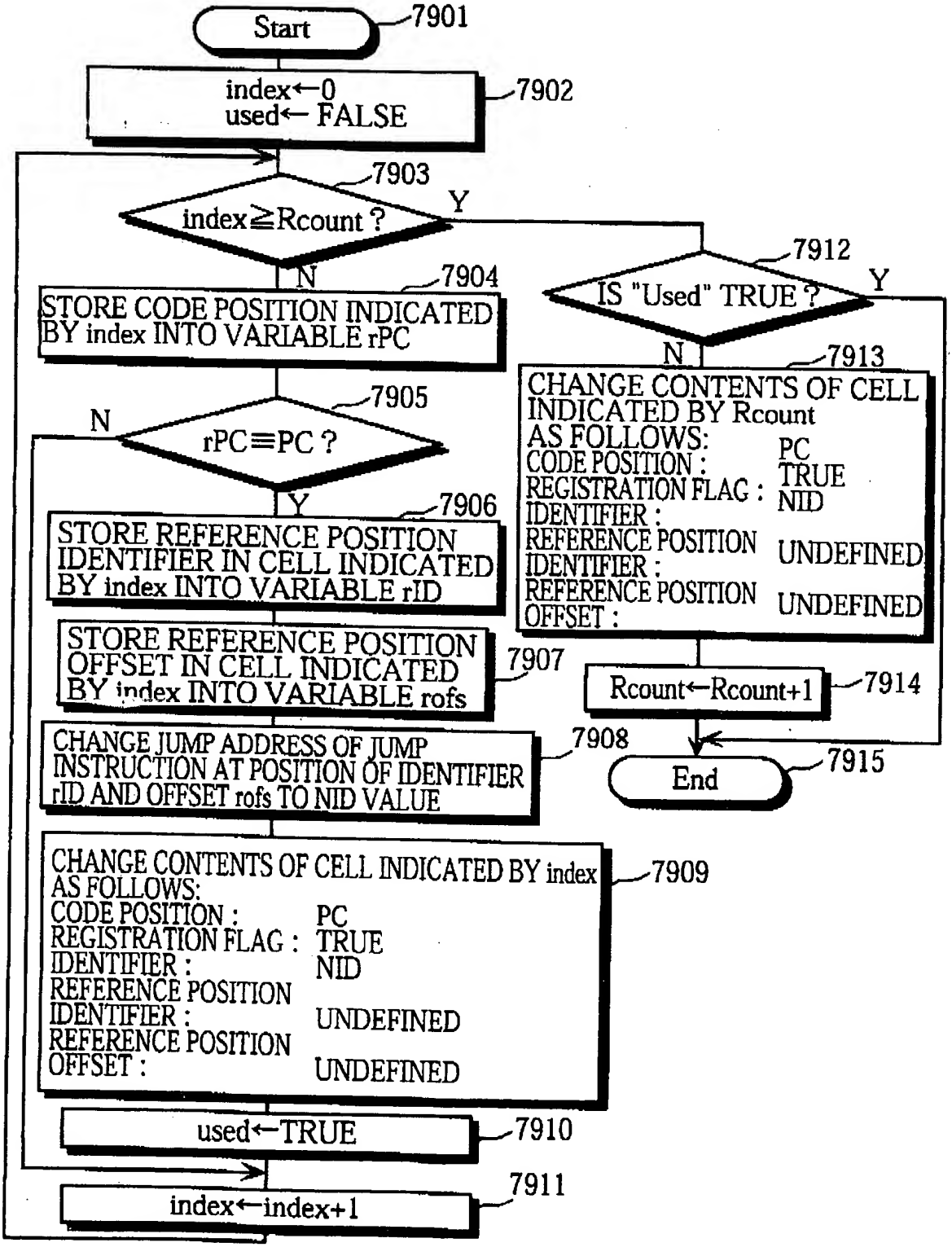

Start
7901
index←0
used← FALSE
7902
7903
index≧Rcount ?
Y
N
7904
STORE CODE POSITION INDICATED BY index INTO VARIABLE rPC
7905
rPC≡PC ?
N
Y
7906
STORE REFERENCE POSITION IDENTIFIER IN CELL INDICATED BY index INTO VARIABLE rID
STORE REFERENCE POSITION OFFSET IN CELL INDICATED BY index INTO VARIABLE rofs
7907
CHANGE JUMP ADDRESS OF JUMP INSTRUCTION AT POSITION OF IDENTIFIER rID AND OFFSET rofs TO NID VALUE
7908
CHANGE CONTENTS OF CELL INDICATED BY index AS FOLLOWS:
CODE POSITION : PC
REGISTRATION FLAG : TRUE
IDENTIFIER : NID
REFERENCE POSITION IDENTIFIER : UNDEFINED
REFERENCE POSITION OFFSET : UNDEFINED
7909
used←TRUE
7910
index←index+1
7911
7912
IS "Used" TRUE ?
Y
N
7913
CHANGE CONTENTS OF CELL INDICATED BY Rcount AS FOLLOWS:
CODE POSITION : PC
REGISTRATION FLAG : TRUE
IDENTIFIER : NID
REFERENCE POSITION IDENTIFIER : UNDEFINED
REFERENCE POSITION OFFSET : UNDEFINED
Rcount←Rcount+1
7914
End
7915

FIG. 82

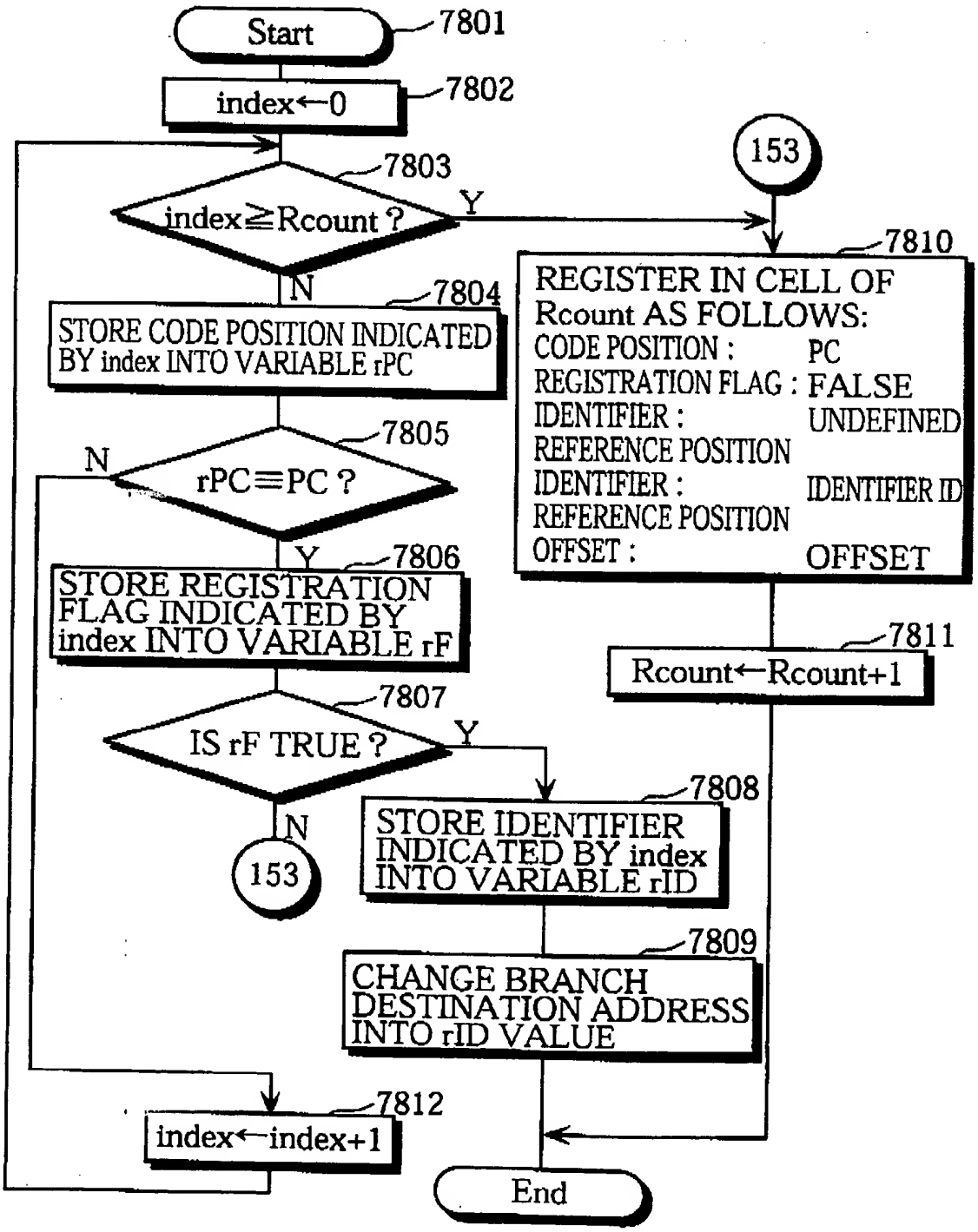

Start
7801
index←0
7802
7803
index≧Rcount ?
Y
N
7804
STORE CODE POSITION INDICATED BY index INTO VARIABLE rPC
7805
rPC≡PC ?
N
Y
7806
STORE REGISTRATION FLAG INDICATED BY index INTO VARIABLE rF
7807
IS rF TRUE ?
Y
N
153
153
7808
STORE IDENTIFIER INDICATED BY index INTO VARIABLE rID
7809
CHANGE BRANCH DESTINATION ADDRESS INTO rID VALUE
7810
REGISTER IN CELL OF Rcount AS FOLLOWS:
CODE POSITION : PC
REGISTRATION FLAG : FALSE
IDENTIFIER : UNDEFINED
REFERENCE POSITION IDENTIFIER : IDENTIFIER ID
REFERENCE POSITION OFFSET : OFFSET
7811
Rcount←Rcount+1
7812
index←index+1
End

FIG. 83

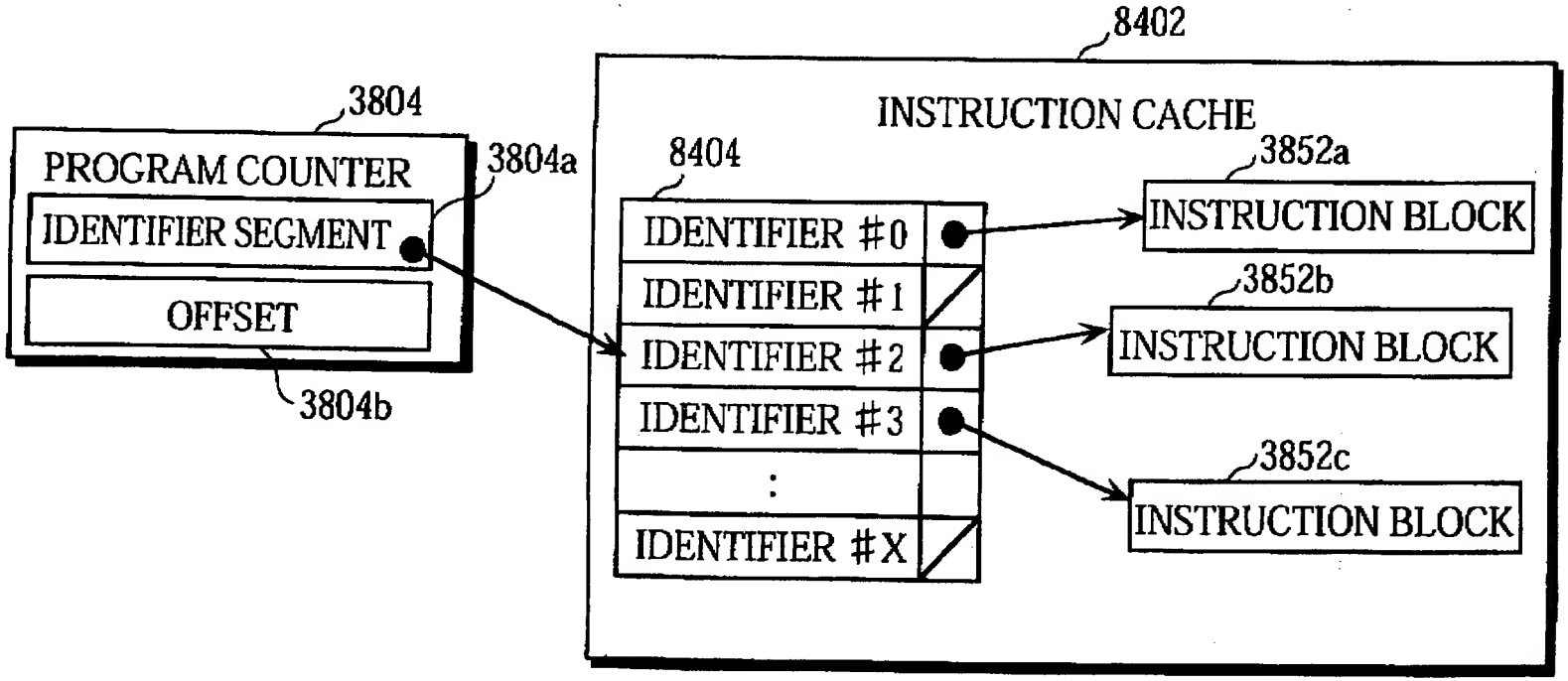

8402
INSTRUCTION CACHE
3804
PROGRAM COUNTER
3804a
IDENTIFIER SEGMENT
OFFSET
3804b
8404
IDENTIFIER #0
IDENTIFIER #1
IDENTIFIER #2
IDENTIFIER #3
:
IDENTIFIER #X
3852a
INSTRUCTION BLOCK
3852b
INSTRUCTION BLOCK
3852c
INSTRUCTION BLOCK

FIG. 84

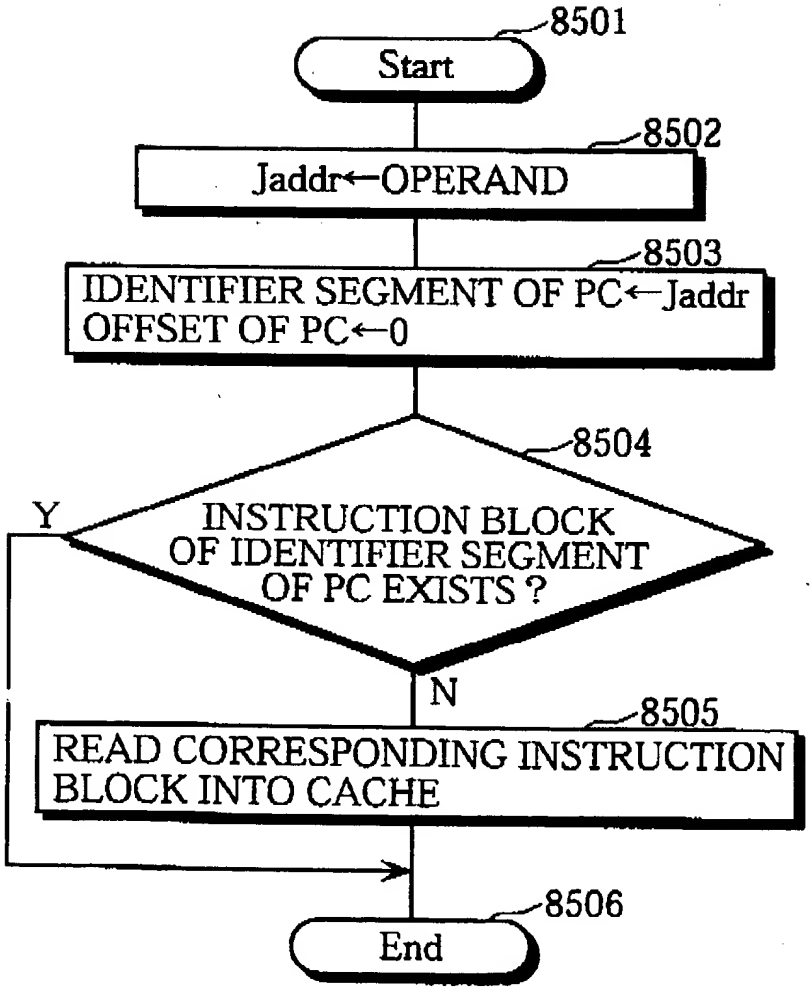

8501
Start
8502
Jaddr←OPERAND
8503
IDENTIFIER SEGMENT OF PC←Jaddr
OFFSET OF PC←0
8504
INSTRUCTION BLOCK OF IDENTIFIER SEGMENT OF PC EXISTS ?
Y
N
8505
READ CORRESPONDING INSTRUCTION BLOCK INTO CACHE
8506
End

FIG. 85

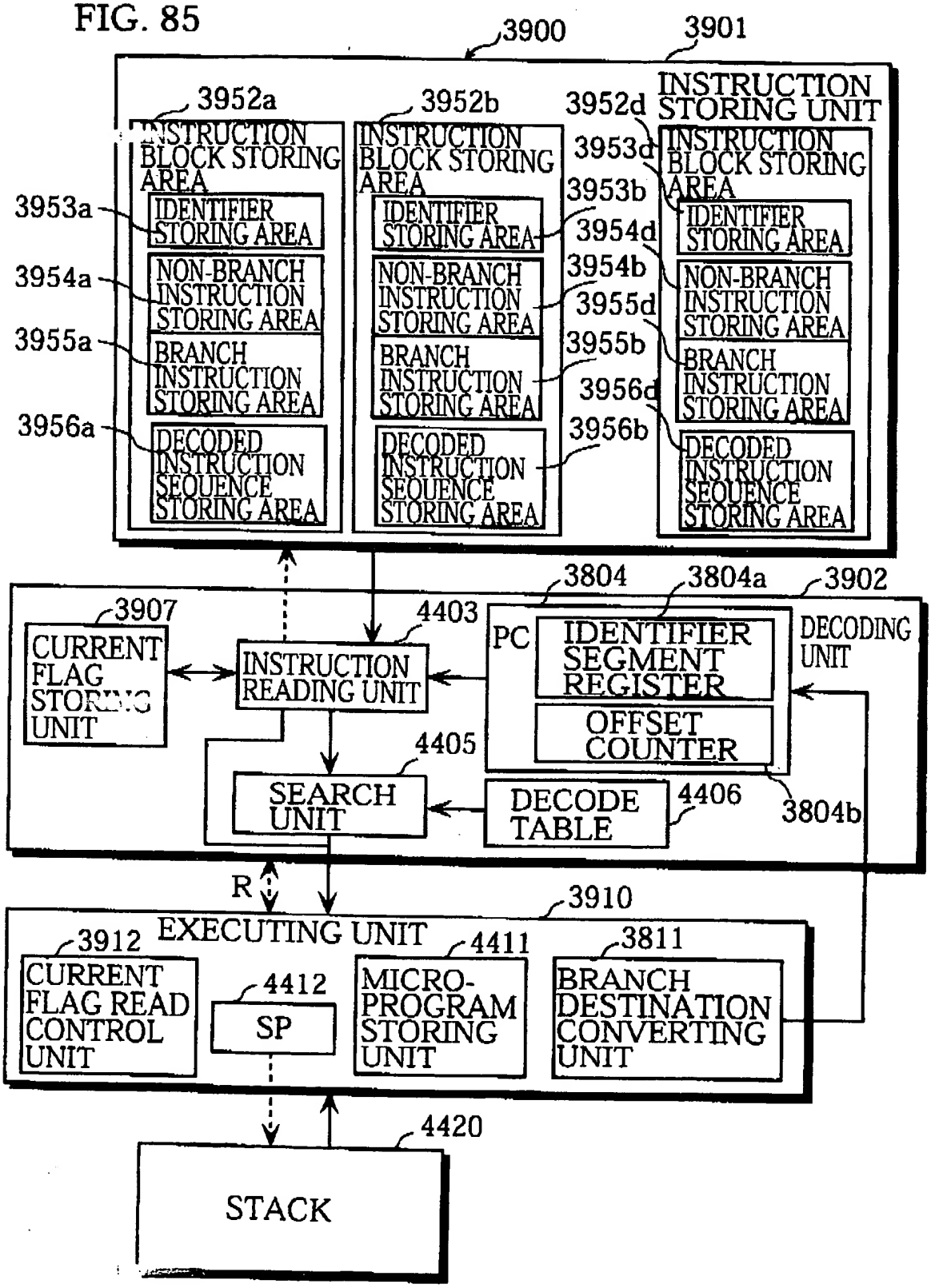

3900
3901
INSTRUCTION STORING UNIT
3952a
3952b
3952d
INSTRUCTION BLOCK STORING AREA
IDENTIFIER STORING AREA
NON-BRANCH INSTRUCTION STORING AREA
BRANCH INSTRUCTION STORING AREA
DECODED INSTRUCTION SEQUENCE STORING AREA
3953a
3953b
3953d
3954a
3954b
3954d
3955a
3955b
3955d
3956a
3956b
3956d
3804
3804a
3902
3907
4403
CURRENT FLAG STORING UNIT
INSTRUCTION READING UNIT
PC
IDENTIFIER SEGMENT REGISTER
OFFSET COUNTER
DECODING UNIT
4405
SEARCH UNIT
DECODE TABLE
4406
3804b
R
3910
EXECUTING UNIT
3912
4411
3811
CURRENT FLAG READ CONTROL UNIT
4412
SP
MICRO-PROGRAM STORING UNIT
BRANCH DESTINATION CONVERTING UNIT
4420
STACK

FIG. 86A

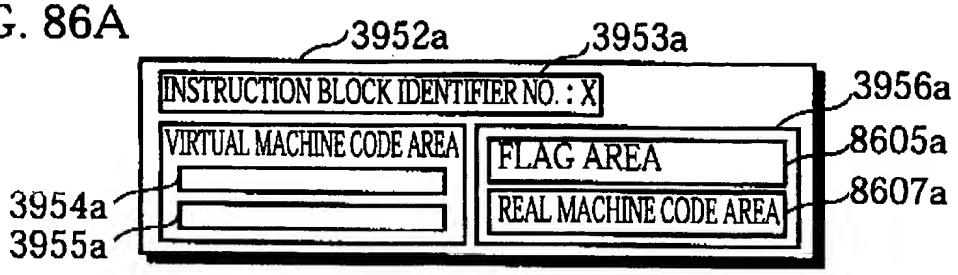


FIG. 86B

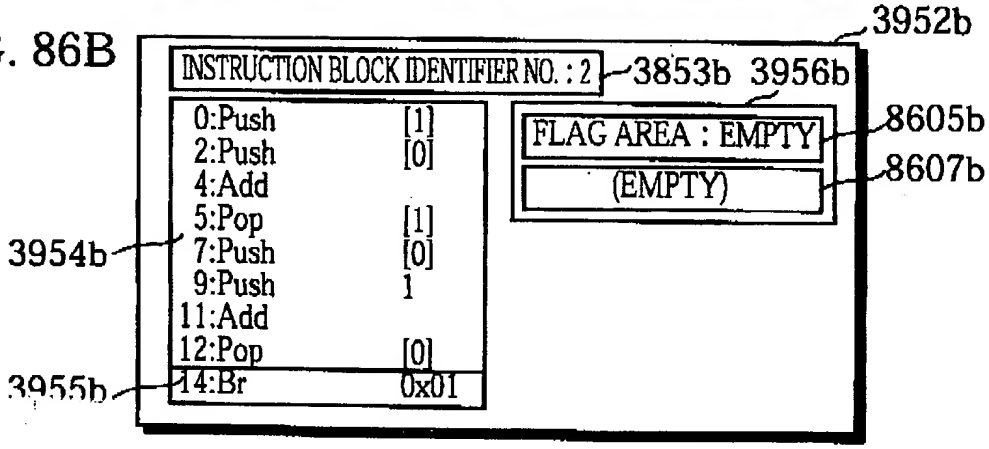


FIG. 86C

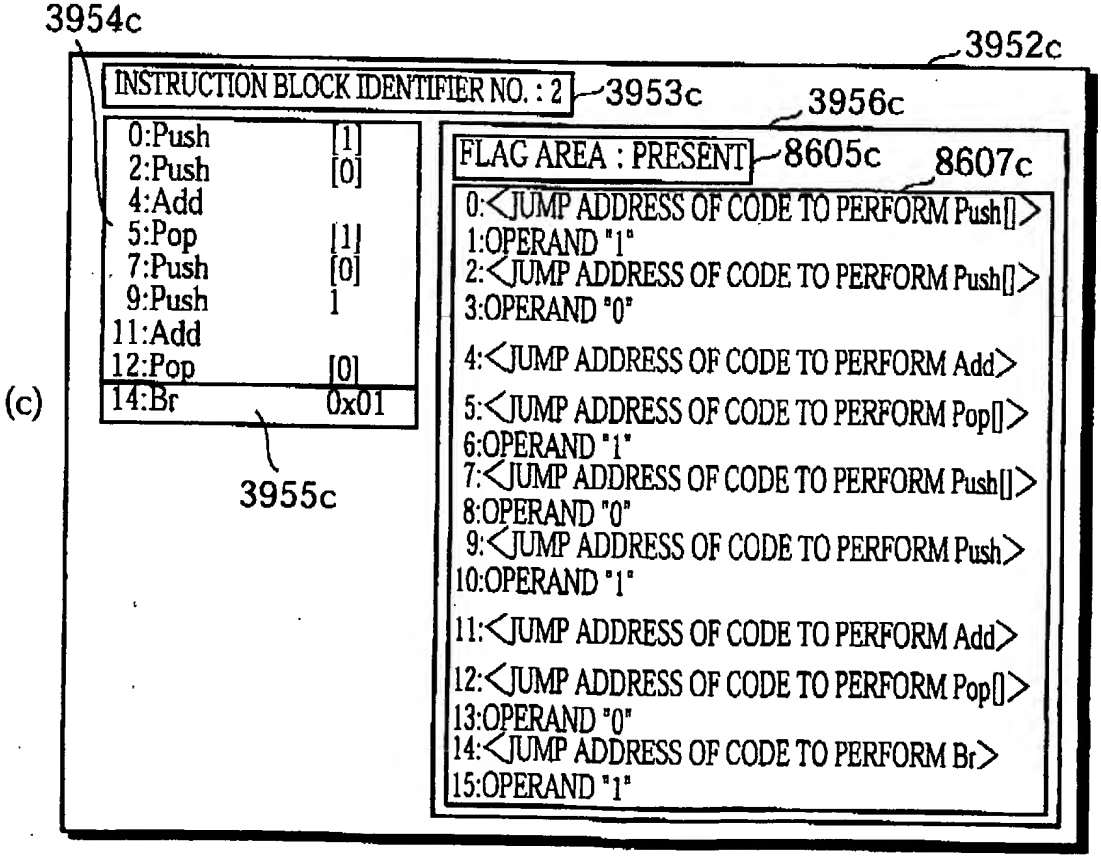


FIG. 87

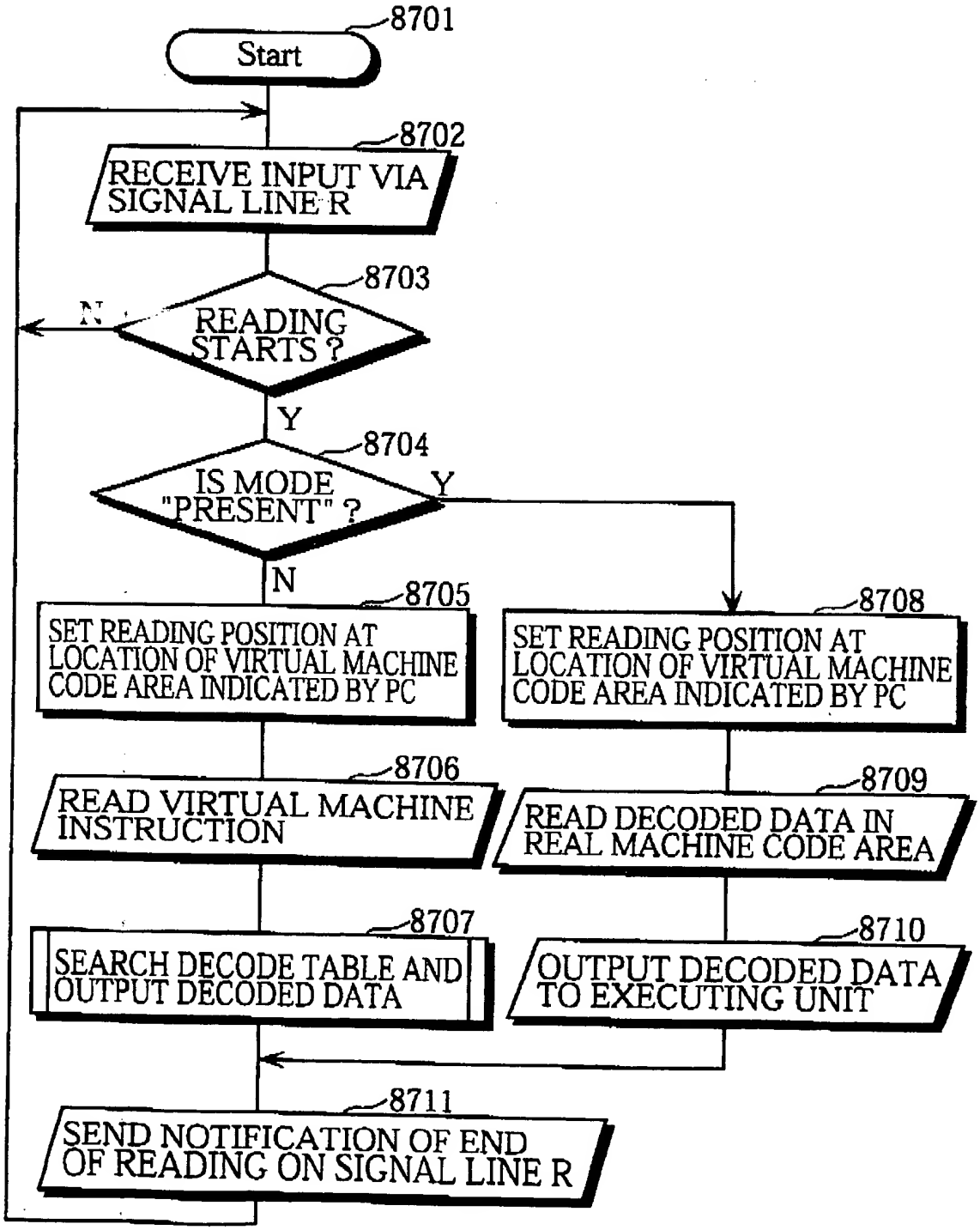

8701
Start
8702
RECEIVE INPUT VIA SIGNAL LINE R
8703
READING STARTS ?
N
Y
8704
IS MODE "PRESENT" ?
Y
N
8705
SET READING POSITION AT LOCATION OF VIRTUAL MACHINE CODE AREA INDICATED BY PC
8706
READ VIRTUAL MACHINE INSTRUCTION
8707
SEARCH DECODE TABLE AND OUTPUT DECODED DATA
8708
SET READING POSITION AT LOCATION OF VIRTUAL MACHINE CODE AREA INDICATED BY PC
8709
READ DECODED DATA IN REAL MACHINE CODE AREA
8710
OUTPUT DECODED DATA TO EXECUTING UNIT
8711
SEND NOTIFICATION OF END OF READING ON SIGNAL LINE R

FIG. 88

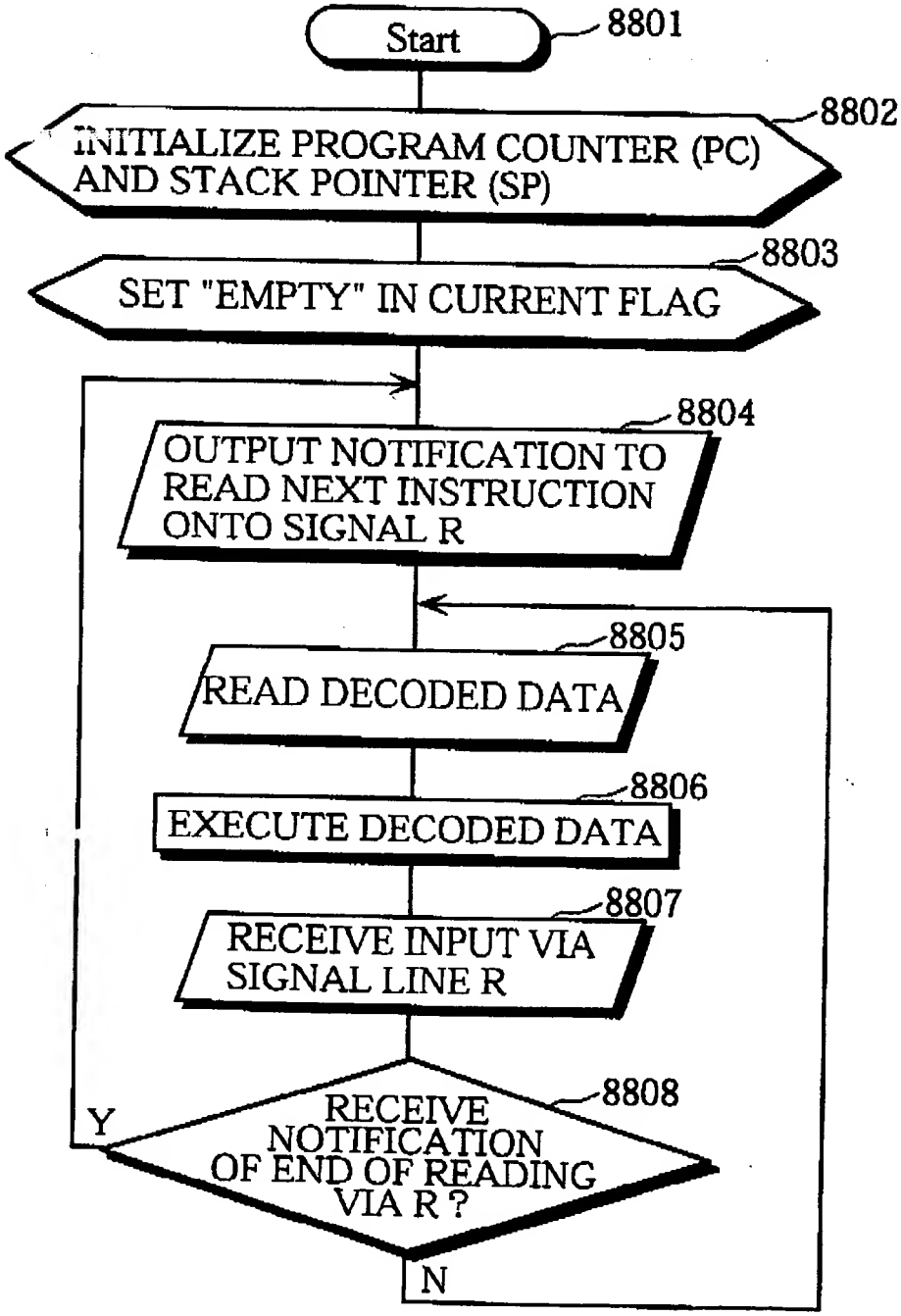

Start
8801
INITIALIZE PROGRAM COUNTER (PC) AND STACK POINTER (SP)
8802
SET "EMPTY" IN CURRENT FLAG
8803
OUTPUT NOTIFICATION TO READ NEXT INSTRUCTION ONTO SIGNAL R
8804
READ DECODED DATA
8805
EXECUTE DECODED DATA
8806
RECEIVE INPUT VIA SIGNAL LINE R
8807
RECEIVE NOTIFICATION OF END OF READING VIA R ?
8808
Y
N

FIG. 89

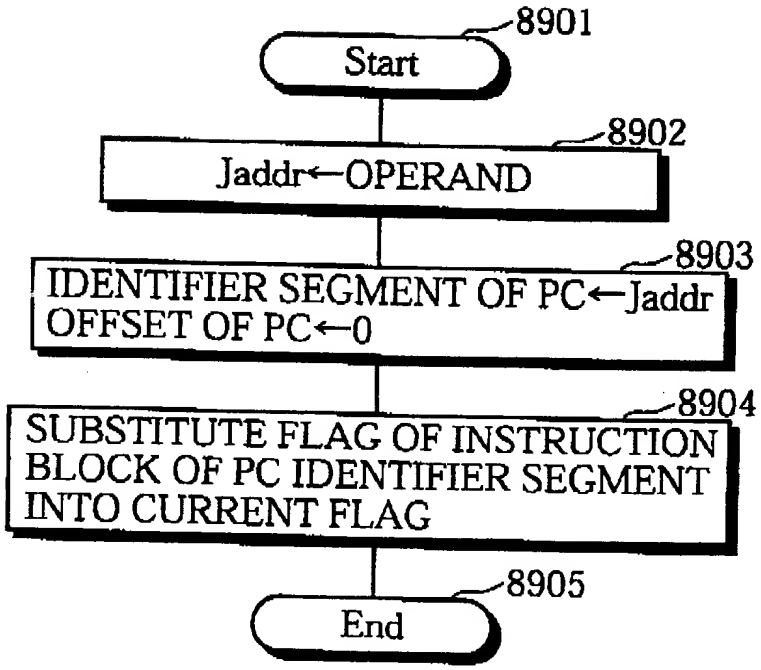

8901
Start
8902
Jaddr←OPERAND
8903
IDENTIFIER SEGMENT OF PC←Jaddr
OFFSET OF PC←0
8904
SUBSTITUTE FLAG OF INSTRUCTION BLOCK OF PC IDENTIFIER SEGMENT INTO CURRENT FLAG
8905
End

FIG. 90

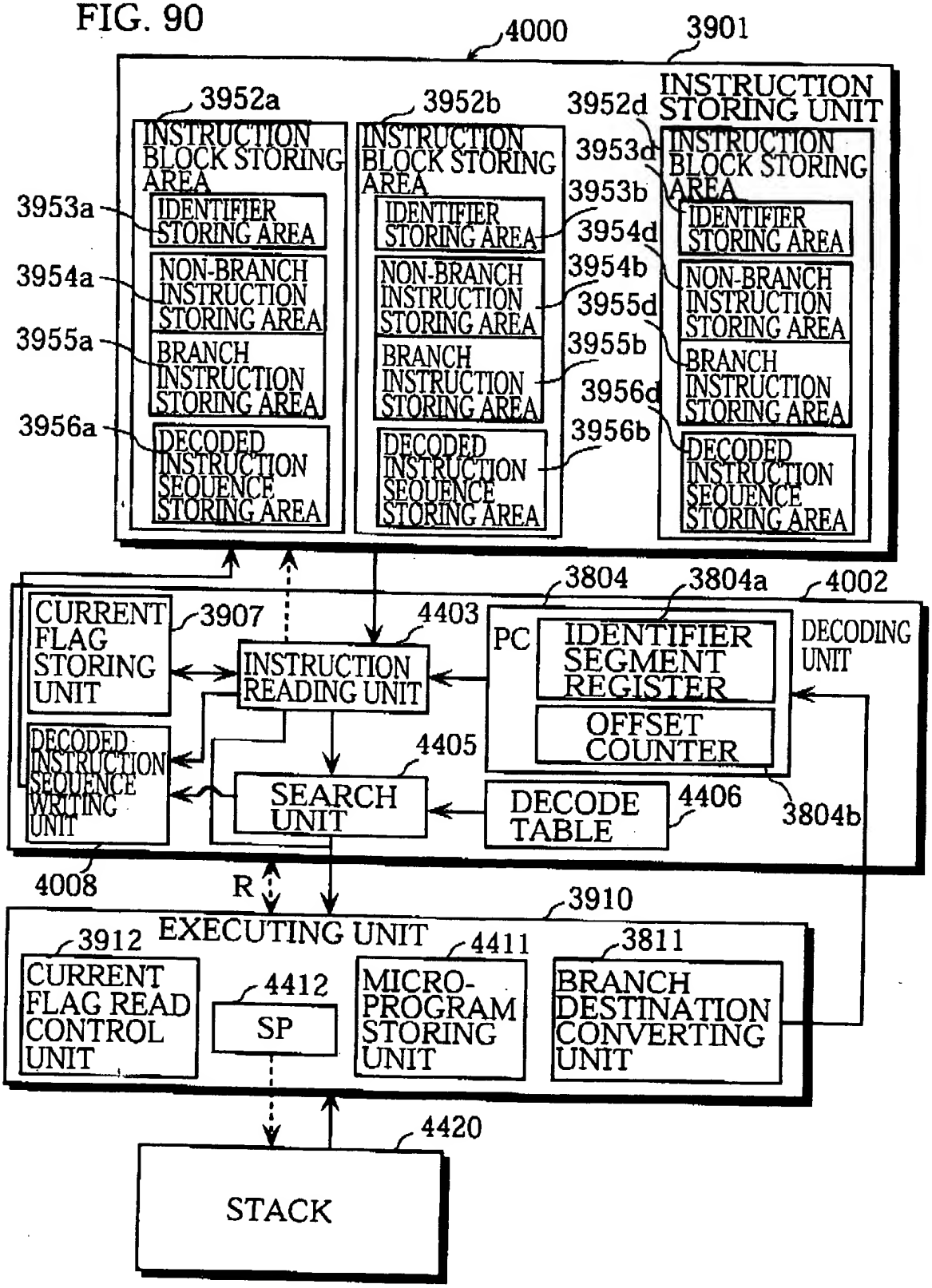

4000
3901
INSTRUCTION STORING UNIT
3952a
3952b
3952d
INSTRUCTION BLOCK STORING AREA
IDENTIFIER STORING AREA
NON-BRANCH INSTRUCTION STORING AREA
BRANCH INSTRUCTION STORING AREA
DECODED INSTRUCTION SEQUENCE STORING AREA
3953a
3953b
3953d
3954a
3954b
3954d
3955a
3955b
3955d
3956a
3956b
3956d
3804
3804a
4002
CURRENT FLAG STORING UNIT
3907
4403
INSTRUCTION READING UNIT
PC
IDENTIFIER SEGMENT REGISTER
OFFSET COUNTER
DECODING UNIT
DECODED INSTRUCTION SEQUENCE WRITING UNIT
4405
SEARCH UNIT
DECODE TABLE
4406
3804b
4008
R
3910
EXECUTING UNIT
3912
4411
3811
CURRENT FLAG READ CONTROL UNIT
4412
SP
MICRO-PROGRAM STORING UNIT
BRANCH DESTINATION CONVERTING UNIT
4420
STACK

FIG. 91

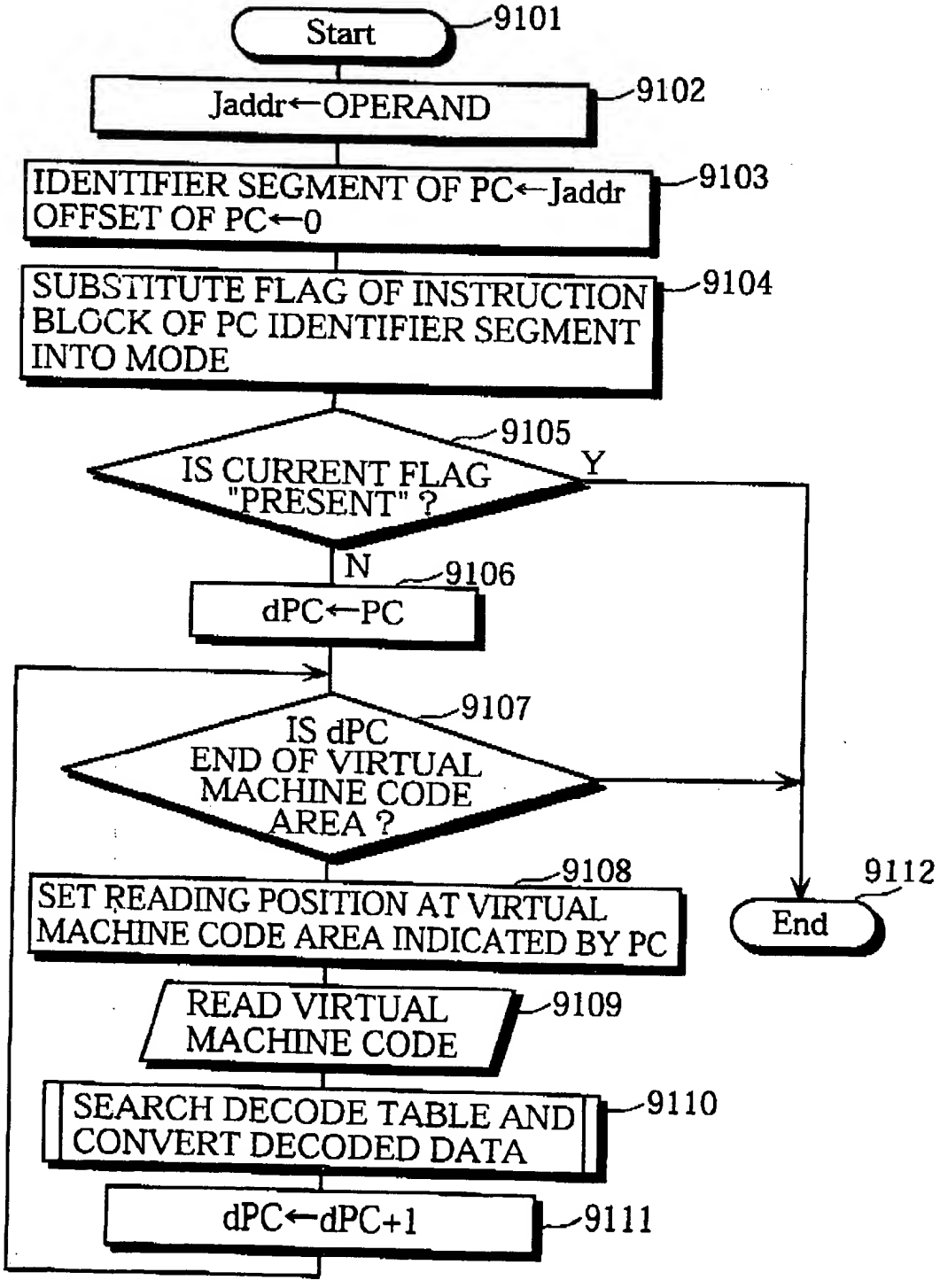

Start
9101
Jaddr←OPERAND
9102
IDENTIFIER SEGMENT OF PC←Jaddr
OFFSET OF PC←0
9103
SUBSTITUTE FLAG OF INSTRUCTION BLOCK OF PC IDENTIFIER SEGMENT INTO MODE
9104
IS CURRENT FLAG "PRESENT" ?
9105
Y
N
dPC←PC
9106
IS dPC END OF VIRTUAL MACHINE CODE AREA ?
9107
SET READING POSITION AT VIRTUAL MACHINE CODE AREA INDICATED BY PC
9108
READ VIRTUAL MACHINE CODE
9109
SEARCH DECODE TABLE AND CONVERT DECODED DATA
9110
dPC←dPC+1
9111
End
9112

FIG. 92

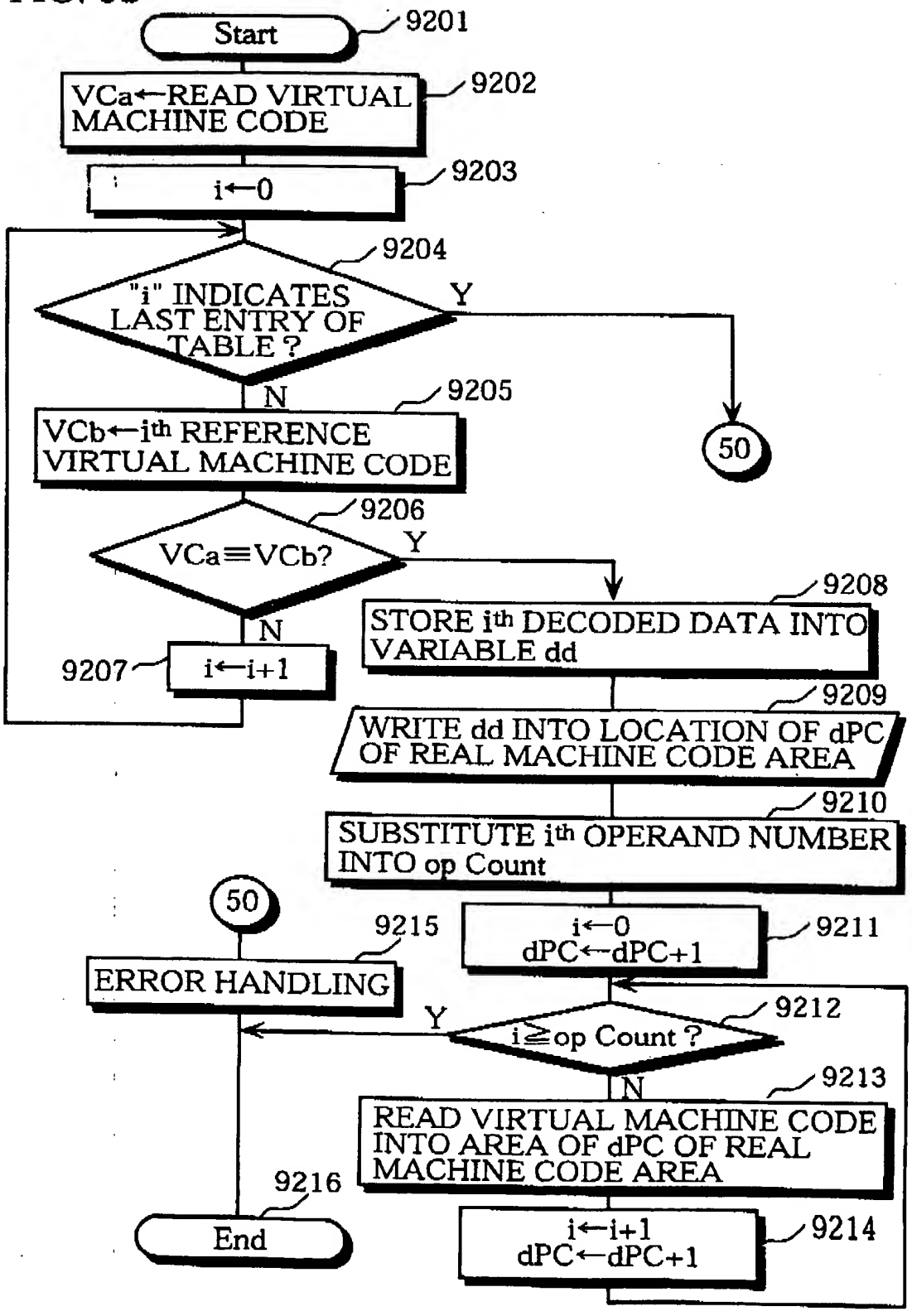

Start
9201
VCa←READ VIRTUAL MACHINE CODE
9202
i←0
9203
"i" INDICATES LAST ENTRY OF TABLE ?
9204
Y
N
50
VCb←ith REFERENCE VIRTUAL MACHINE CODE
9205
VCa≡VCb?
9206
Y
N
STORE ith DECODED DATA INTO VARIABLE dd
9208
9207
i←i+1
WRITE dd INTO LOCATION OF dPC OF REAL MACHINE CODE AREA
9209
SUBSTITUTE ith OPERAND NUMBER INTO op Count
9210
50
i←0
dPC←dPC+1
9211
9215
ERROR HANDLING
i≧op Count ?
9212
Y
N
READ VIRTUAL MACHINE CODE INTO AREA OF dPC OF REAL MACHINE CODE AREA
9213
9216
End
i←i+1
dPC←dPC+1
9214

FIG. 93

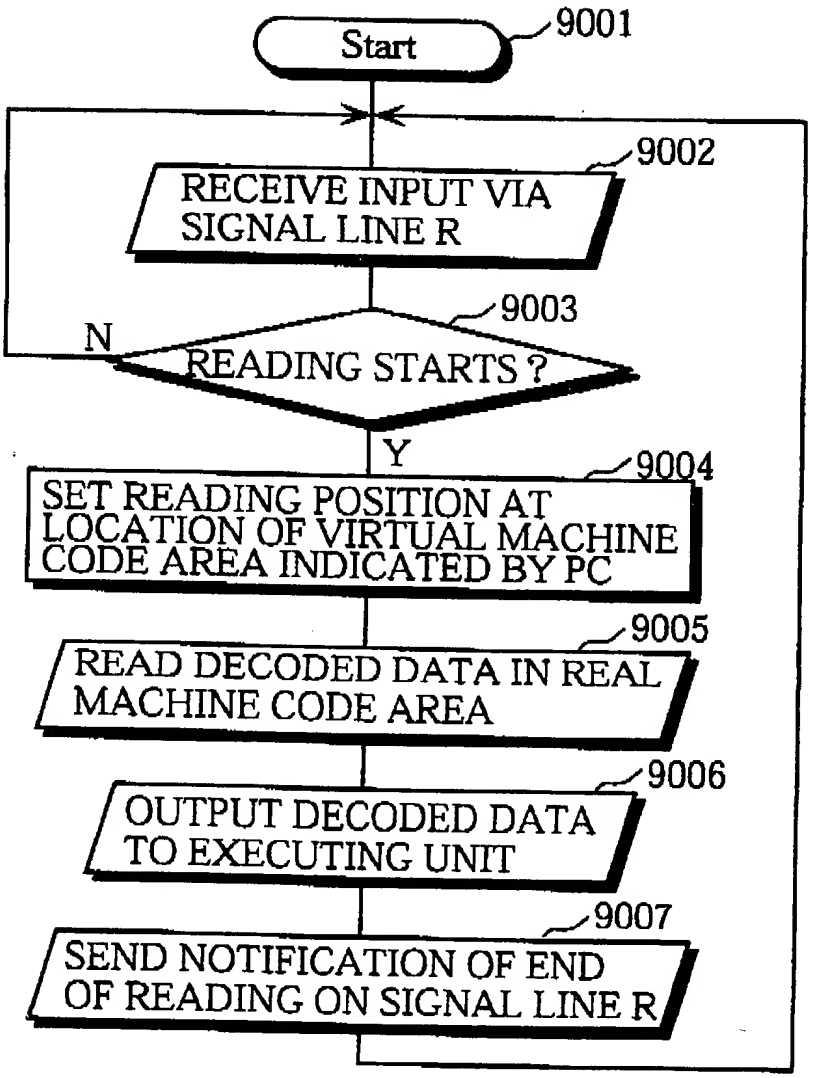

Start
9001
RECEIVE INPUT VIA SIGNAL LINE R
9002
N
READING STARTS ?
9003
Y
SET READING POSITION AT LOCATION OF VIRTUAL MACHINE CODE AREA INDICATED BY PC
9004
READ DECODED DATA IN REAL MACHINE CODE AREA
9005
OUTPUT DECODED DATA TO EXECUTING UNIT
9006
SEND NOTIFICATION OF END OF READING ON SIGNAL LINE R
9007

FIG. 94

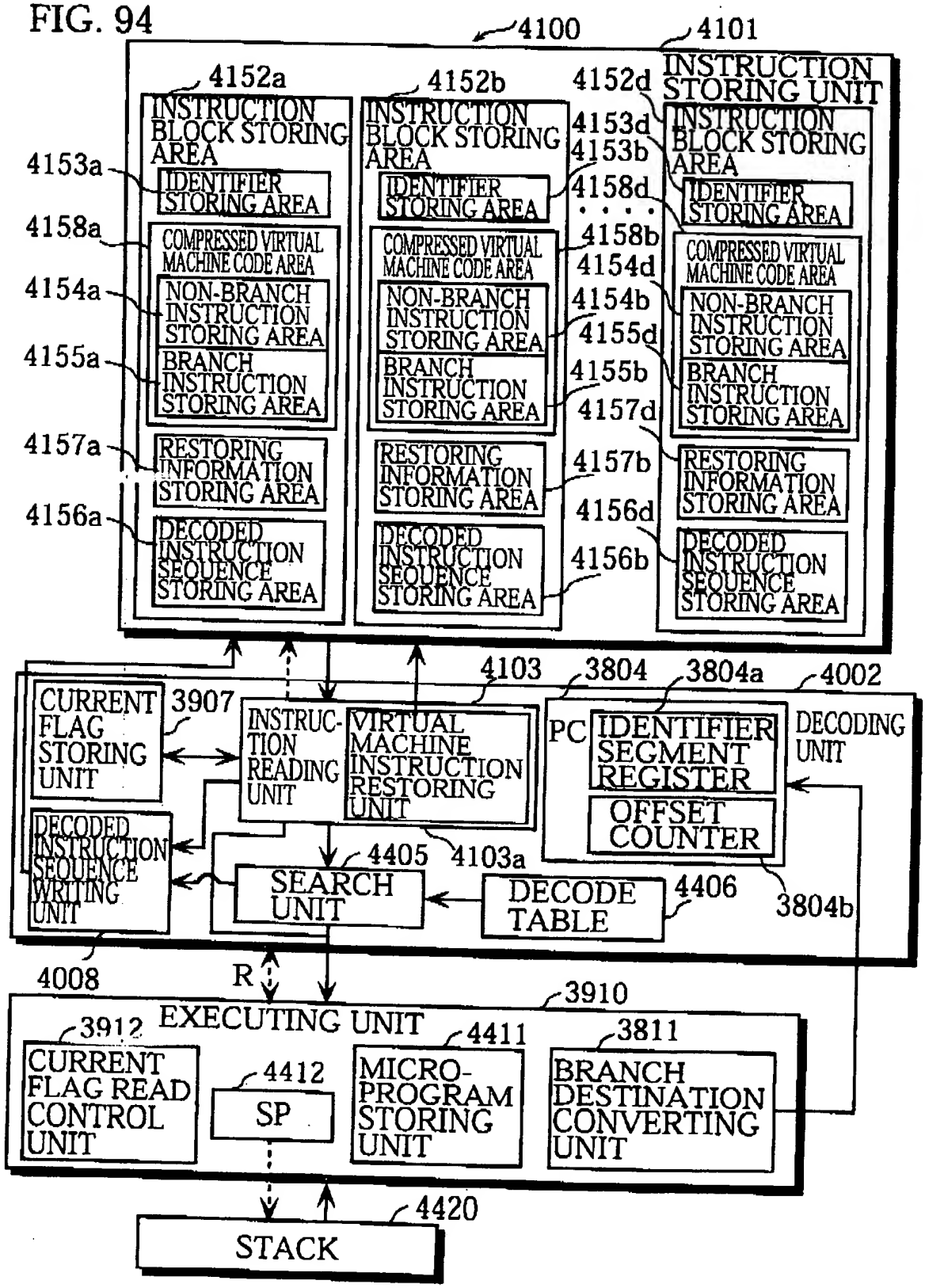

4100
4101
INSTRUCTION STORING UNIT
4152a
INSTRUCTION BLOCK STORING AREA
4152b
INSTRUCTION BLOCK STORING AREA
4152d
INSTRUCTION BLOCK STORING AREA
4153a
IDENTIFIER STORING AREA
4153b
IDENTIFIER STORING AREA
4153d
IDENTIFIER STORING AREA
4158a
COMPRESSED VIRTUAL MACHINE CODE AREA
4158b
COMPRESSED VIRTUAL MACHINE CODE AREA
4158d
COMPRESSED VIRTUAL MACHINE CODE AREA
4154a
NON-BRANCH INSTRUCTION STORING AREA
4154b
NON-BRANCH INSTRUCTION STORING AREA
4154d
NON-BRANCH INSTRUCTION STORING AREA
4155a
BRANCH INSTRUCTION STORING AREA
4155b
BRANCH INSTRUCTION STORING AREA
4155d
BRANCH INSTRUCTION STORING AREA
4157a
RESTORING INFORMATION STORING AREA
4157b
RESTORING INFORMATION STORING AREA
4157d
RESTORING INFORMATION STORING AREA
4156a
DECODED INSTRUCTION SEQUENCE STORING AREA
4156b
DECODED INSTRUCTION SEQUENCE STORING AREA
4156d
DECODED INSTRUCTION SEQUENCE STORING AREA
4103
3804
3804a
4002
CURRENT FLAG STORING UNIT
3907
INSTRUC-TION READING UNIT
VIRTUAL MACHINE INSTRUCTION RESTORING UNIT
PC
IDENTIFIER SEGMENT REGISTER
OFFSET COUNTER
DECODING UNIT
DECODED INSTRUCTION SEQUENCE WRITING UNIT
4405
4103a
SEARCH UNIT
DECODE TABLE
4406
3804b
4008
R
3910
EXECUTING UNIT
3912
CURRENT FLAG READ CONTROL UNIT
4412
SP
4411
MICRO-PROGRAM STORING UNIT
3811
BRANCH DESTINATION CONVERTING UNIT
4420
STACK

FIG. 95A

BIT SEQUENCE	VIRTUAL MACHINE INSTRUCTION
000	Push [0]
100	Push 0
101	Pop [0]
110	Pop [1]
111	Br 0x01
0011	Stop
0100	Add
0110	Brz 0x03
0111	Br 0x02
00100	Push [1]
00101	Push 1
01010	Push 10
01011	Sub

FIG. 95B

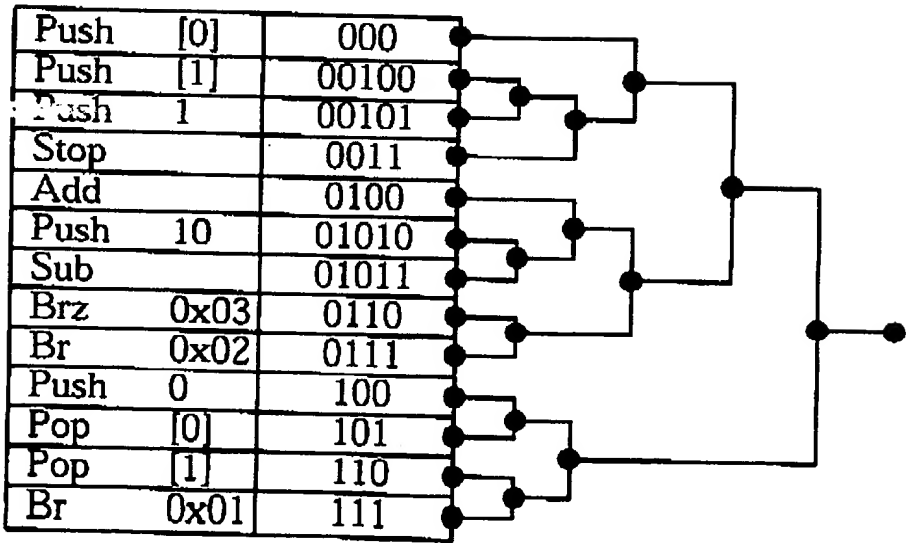


FIG. 96A

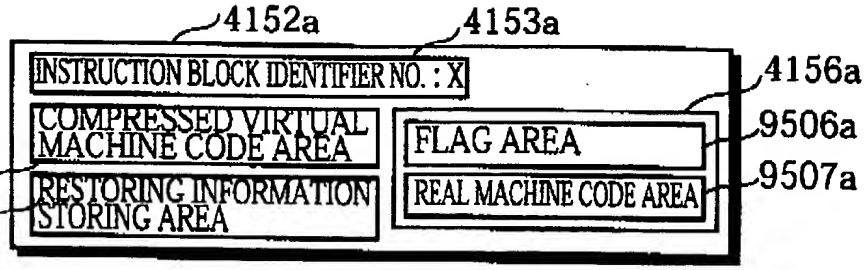


FIG. 96B

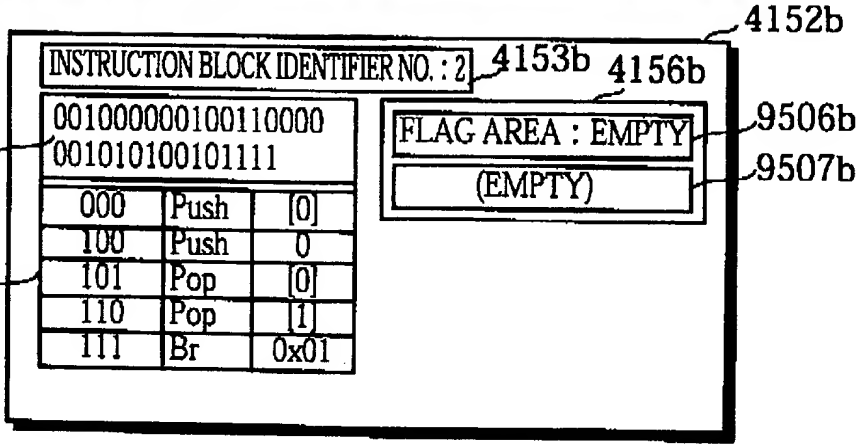


FIG. 96C

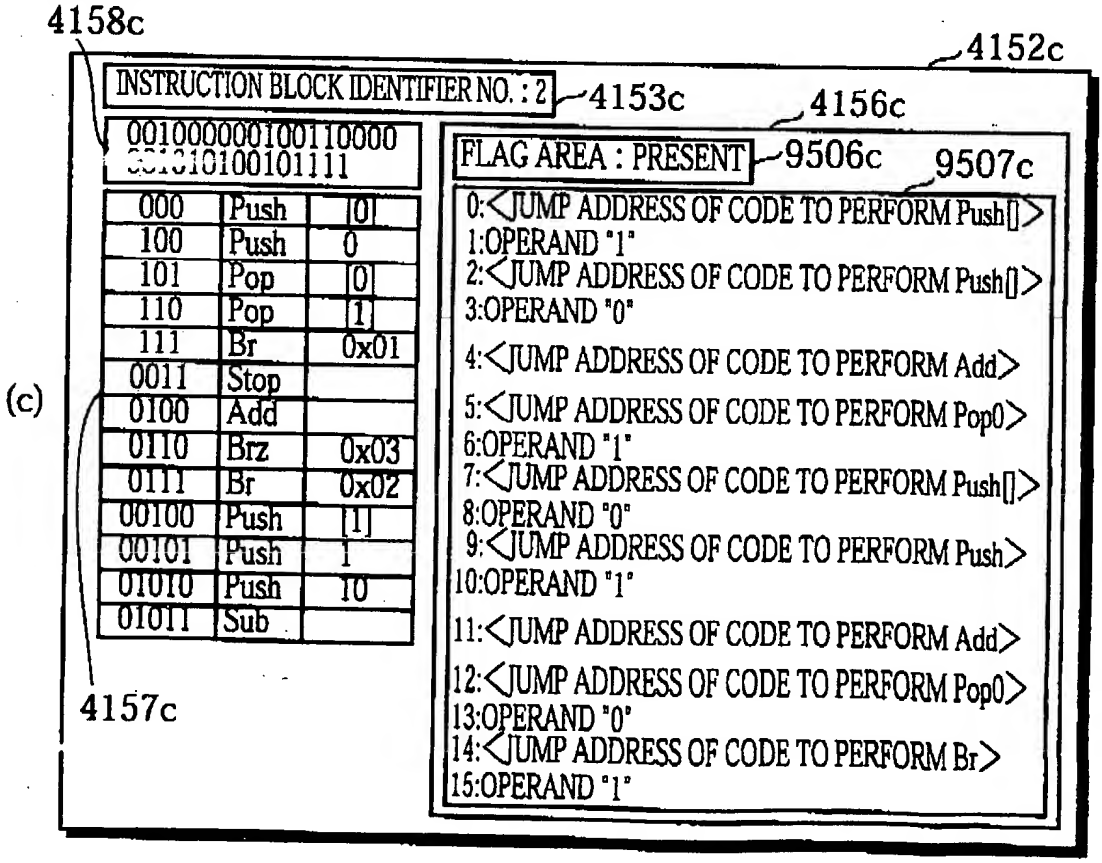


FIG. 97

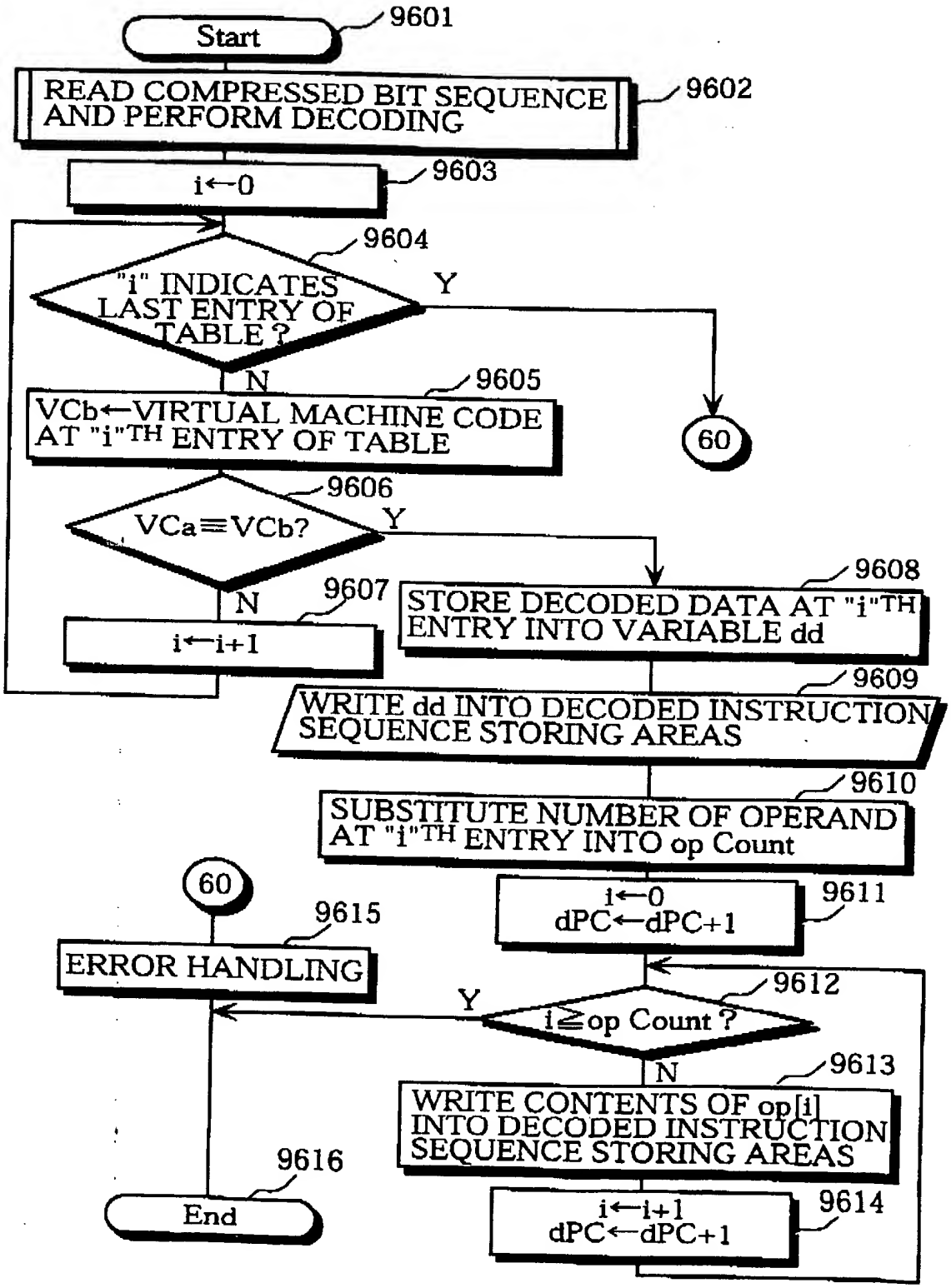

Start
9601
READ COMPRESSED BIT SEQUENCE AND PERFORM DECODING
9602
i←0
9603
"i" INDICATES LAST ENTRY OF TABLE ?
9604
Y
N
VCb←VIRTUAL MACHINE CODE AT "i"TH ENTRY OF TABLE
9605
60
VCa≡VCb?
9606
Y
N
i←i+1
9607
STORE DECODED DATA AT "i"TH ENTRY INTO VARIABLE dd
9608
WRITE dd INTO DECODED INSTRUCTION SEQUENCE STORING AREAS
9609
SUBSTITUTE NUMBER OF OPERAND AT "i"TH ENTRY INTO op Count
9610
i←0
dPC←dPC+1
9611
60
ERROR HANDLING
9615
i≧op Count ?
9612
Y
N
WRITE CONTENTS OF op[i] INTO DECODED INSTRUCTION SEQUENCE STORING AREAS
9613
End
9616
i←i+1
dPC←dPC+1
9614

FIG. 98

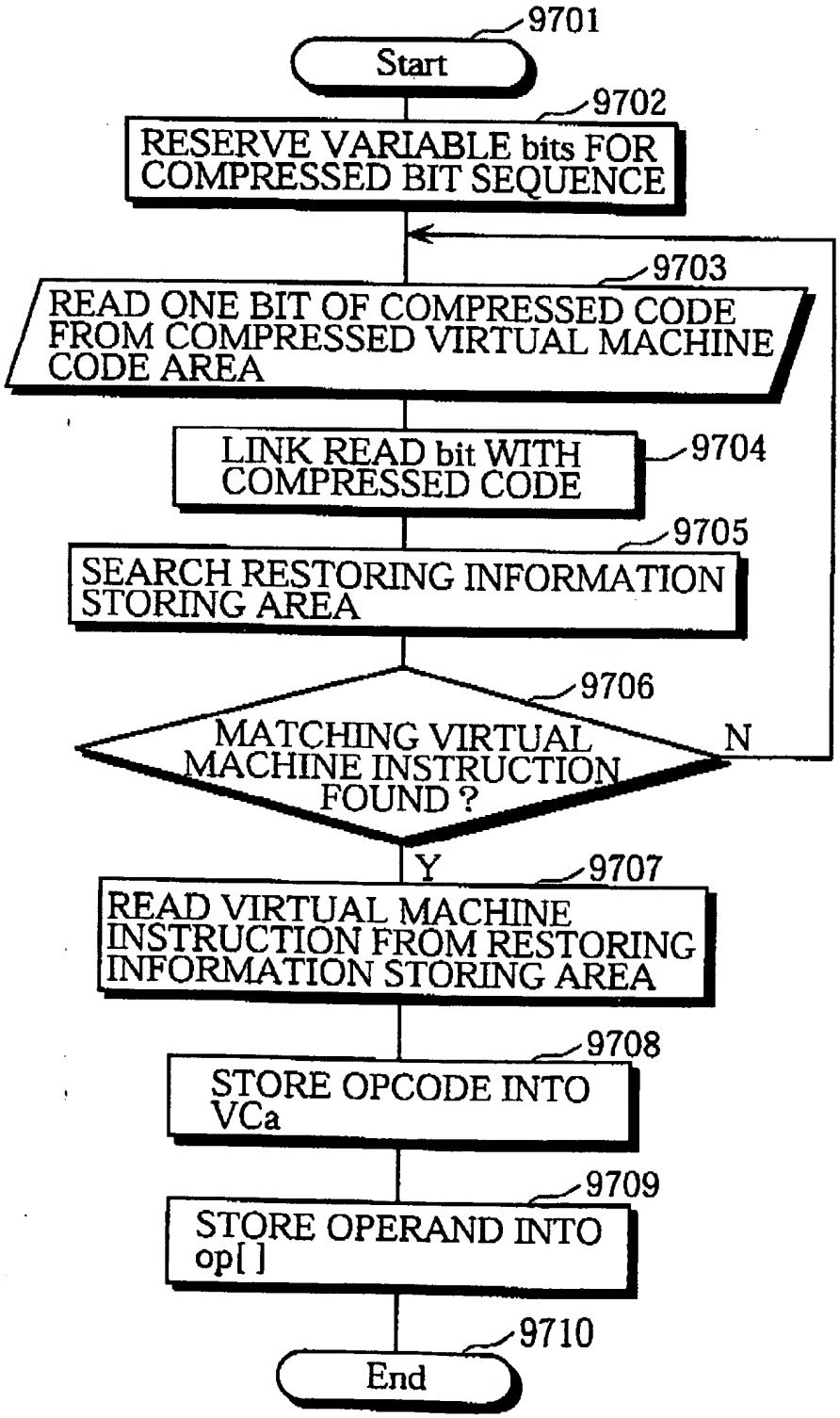

9701
Start
9702
RESERVE VARIABLE bits FOR COMPRESSED BIT SEQUENCE
9703
READ ONE BIT OF COMPRESSED CODE FROM COMPRESSED VIRTUAL MACHINE CODE AREA
LINK READ bit WITH COMPRESSED CODE
9704
9705
SEARCH RESTORING INFORMATION STORING AREA
9706
MATCHING VIRTUAL MACHINE INSTRUCTION FOUND ?
N
Y
9707
READ VIRTUAL MACHINE INSTRUCTION FROM RESTORING INFORMATION STORING AREA
9708
STORE OPCODE INTO VCa
9709
STORE OPERAND INTO op[ ]
9710
End

FIG. 99

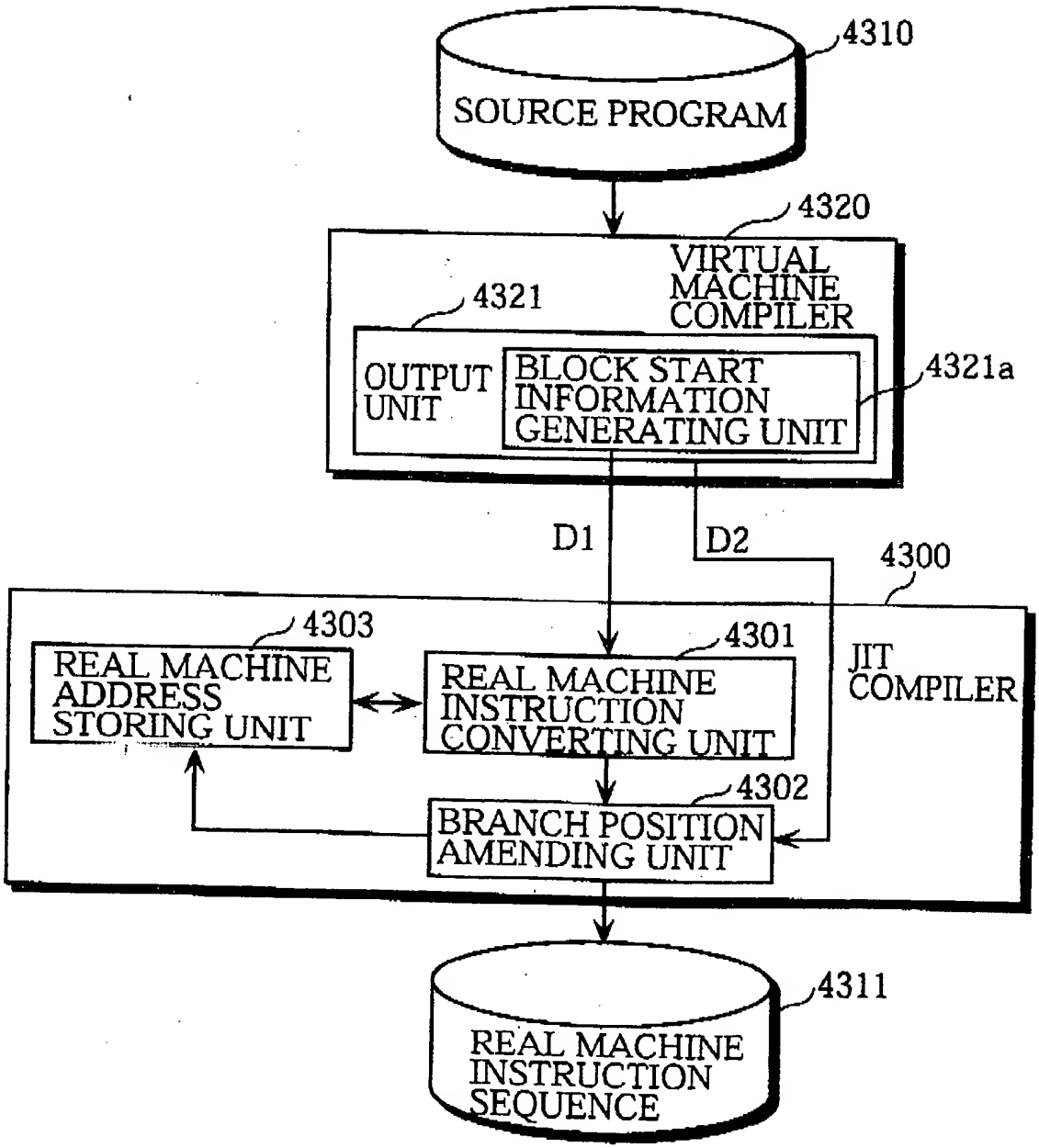

4310
SOURCE PROGRAM
4320
VIRTUAL MACHINE COMPILER
4321
OUTPUT UNIT
BLOCK START INFORMATION GENERATING UNIT
4321a
D1
D2
4300
JIT COMPILER
4303
REAL MACHINE ADDRESS STORING UNIT
4301
REAL MACHINE INSTRUCTION CONVERTING UNIT
4302
BRANCH POSITION AMENDING UNIT
4311
REAL MACHINE INSTRUCTION SEQUENCE

FIG. 100

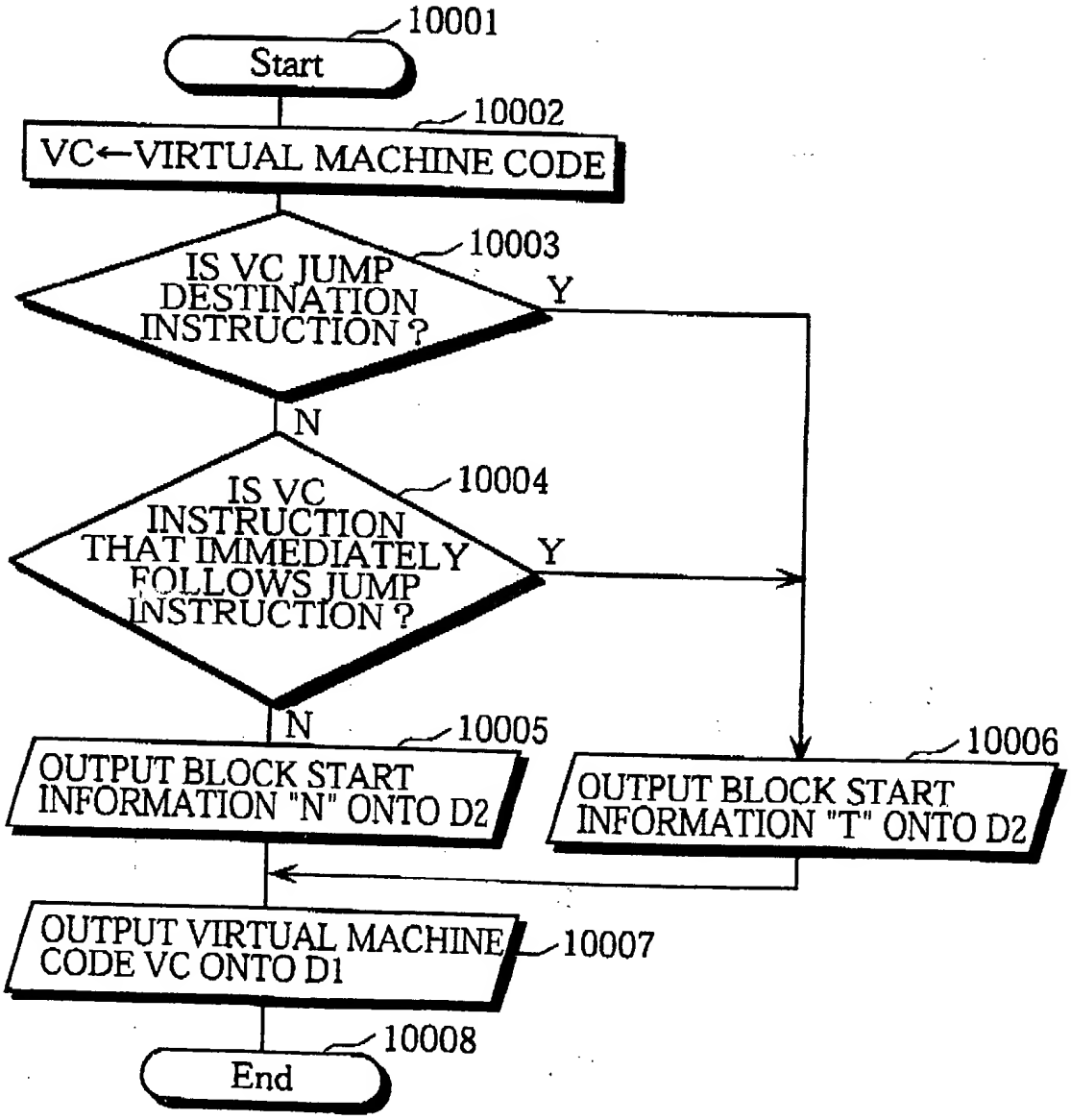

10001
Start
10002
VC←VIRTUAL MACHINE CODE
10003
IS VC JUMP DESTINATION INSTRUCTION ?
Y
N
10004
IS VC INSTRUCTION THAT IMMEDIATELY FOLLOWS JUMP INSTRUCTION ?
Y
N
10005
OUTPUT BLOCK START INFORMATION "N" ONTO D2
10006
OUTPUT BLOCK START INFORMATION "T" ONTO D2
OUTPUT VIRTUAL MACHINE CODE VC ONTO D1
10007
10008
End

FIG. 101

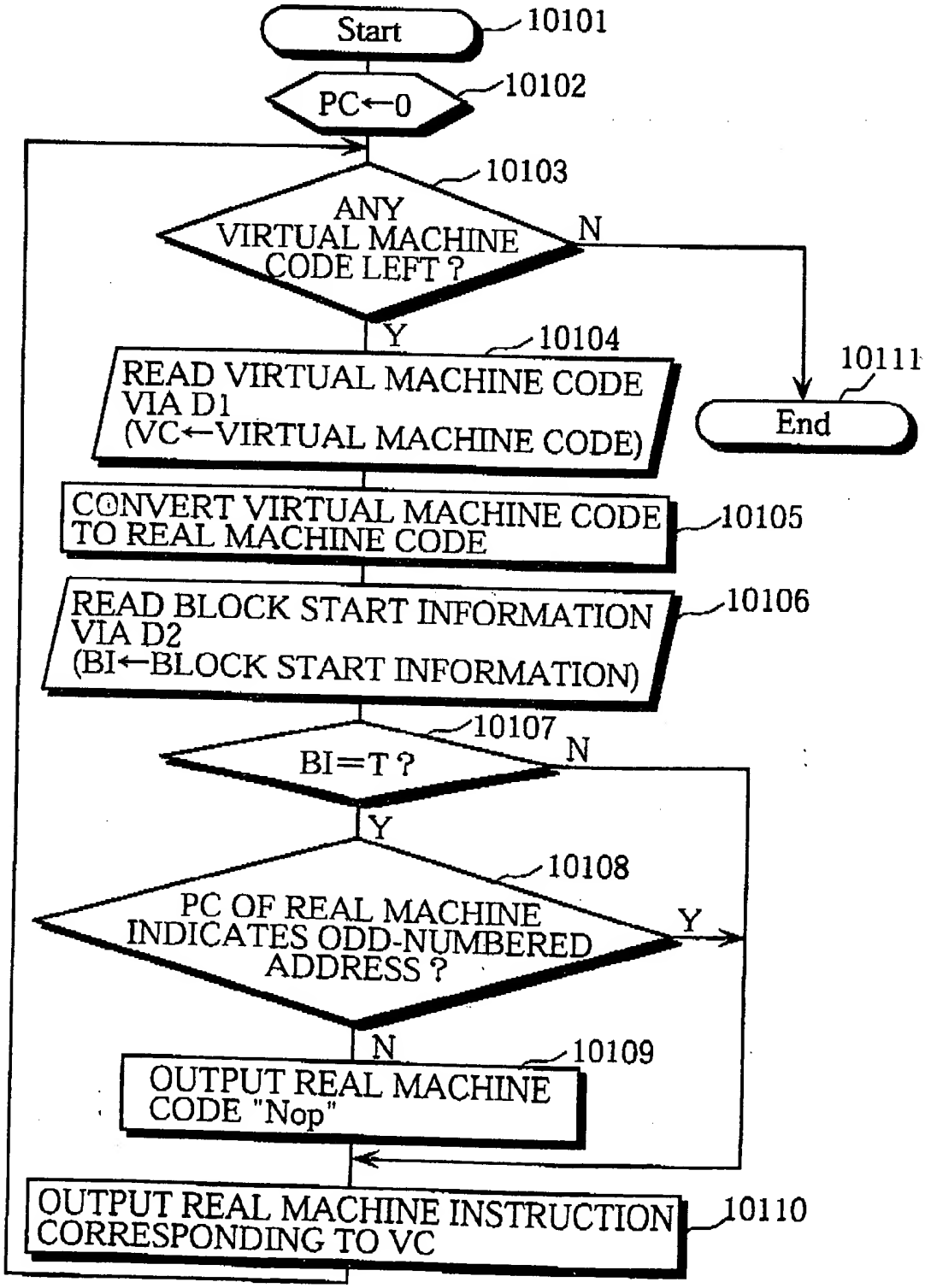

Start
10101
PC←0
10102
ANY VIRTUAL MACHINE CODE LEFT ?
10103
N
Y
READ VIRTUAL MACHINE CODE VIA D1 (VC←VIRTUAL MACHINE CODE)
10104
End
10111
CONVERT VIRTUAL MACHINE CODE TO REAL MACHINE CODE
10105
READ BLOCK START INFORMATION VIA D2 (BI←BLOCK START INFORMATION)
10106
BI=T ?
10107
N
Y
PC OF REAL MACHINE INDICATES ODD-NUMBERED ADDRESS ?
10108
Y
N
OUTPUT REAL MACHINE CODE "Nop"
10109
OUTPUT REAL MACHINE INSTRUCTION CORRESPONDING TO VC
10110

FIG. 102

ADDRESS	VIRTUL MACHINE CODE	REAL MACHINE CODE SIZE	CORRESPONDING REAL MACHINE CODE ADDRESS	BLOCK START INFORMATION	Nop OUTPUT
0	Push 0	4	0-3	T	-
2	Pop [0]	5	4-8	N	-
4	Push 0	4	9-12	N	-
6	Pop [1]	5	13-17	N	-
8	Push [0]	5	18-22	T	-
10	Push 10	4	23-26	N	-
12	Sub	3	27-29	N	-
13	Brz 31	5	30-34	N	Nop
15	Push [1]	5	36-40	T	-
17	Push [0]	5	41-45	N	-
19	Add	3	46-48	N	-
20	Pop [1]	5	49-53	N	-
22	Push [0]	5	54-58	N	-
24	Push 1	4	59-62	N	-
26	Add	3	63-65	N	-
27	Pop [0]	5	66-70	N	-
29	Br 8	3	71-73	N	-
31	Stop	2	74-75	T	-

FIG. 103